KU LEUVEN

FACULTY OF ENGINEERING TECHNOLOGY

TECHNOLOGY CAMPUS GEEL

Upgrading the Interface and Developer Tools of the Trigger Supervisor Software Framework of the CMS experiment at CERN

Glenn DIRKX

Supervisor: Peter Karsmakers

Co-supervisor: Christos Lazaridis

Master Thesis submitted to obtain the degree of
Master of Science in Engineering Technology:
Master of Science in Electronics Engineering
Internet Computing

Academic Year 2015 - 2016

This page is intentionally left almost blank

This page is intentionally left almost blank

Acknowledgements

I would like to thank the following people for their assistance during this project:

Christos Lazaridis for being a great mentor and for not getting mad when I break the nightlies or even SVN itself.

Alessandro Thea for his advice on how to proceed with implementing new functionalities and his supply of motivation and inspiration.

Evangelos Paradas for his guidance trough the architecture of the TS and pointing me to useful resources.

Simone Bologna for his enthusiasm and patience when finding bugs, and his steady supply of ideas.

Furthermore I would like to express my thanks to the entire Online Software team for the freedom and trust I've been given that allowed this project to get as far as it has.

This page is intentionally left almost blank

Abstract

The Compact Muon Solenoid (CMS) Trigger Supervisor (TS) is a software framework that has been designed to handle the CMS Level-1 trigger setup, configuration and monitoring during data taking as well as all communications with the main run control of CMS.

The interface consists of a web-based GUI rendered by a back-end C++ framework (AjaXell) and a front-end JavaScript framework (Dojo). These provide developers with the tools they need to to write their own custom control panels.

However, currently there is much frustration with this framework given the age of the Dojo library and the various hacks needed to implement modern use cases.

The task at hand is to renew this library and its developer tools, updating it to use the newest standards and technologies, while maintaining full compatibility with legacy code.

This document describes the requirements, development process, and changes to this framework that were included in the upgrade from v2.x to v3.x.

Keywords: CERN, CMS, L1 Trigger, C++, Polymer, Web Components.

This page is intentionally left almost blank

Contents

List of Figures

List of Tables

This page is intentionally left almost blank

List of Symbols

Acronyms

CERN	European Organization for Nuclear Research
LHC	Large Hadron Collider
CMS	Compact Muon Solenoid
ATLAS	A Toroidal LHC ApparatuS
ALICE	A Large Ion Collider Experiment
LHCB	Large Hadron Collider beauty Experiment
QCD	Quantum ChromoDynamics
ECAL	Electromagnetic Calorimeter
HCAL	Hadron Calorimeter
L1	Level-1
L1A	Level-1 Accept
TPG	Trigger Primitive Generator
FPGA	Field-Programmable Gate Array
TS	Trigger Supervisor
FSM	Finite State Machine
ESR	Extended Support Release
XDAQ	Cross Data Acquisition
XAAS	XDAQ as a Service
VM	Virtual Machine
W3C	World Wide Web Consortium
RERO	Release Early, Release Often
ES6	ECMAScript 6, also known as ECMAScript 2015
AJAX	Asynchronous JavaScript And XML
SLC	Scientific Linux CERN
SVN	Apache Subversion
SEO	Search Engine Optimization

This page is intentionally left almost blank

Chapter 1

The Compact Muon Solenoid experiment

The Compact Muon Solenoid (CMS) experiment is one of the main particle detectors at the LHC currently in use at CERN, along with ATLAS, ALICE, and LHCb.

It is a general purpose detector, i.e., it is designed to observe all particle interactions in a collision. CMS is also designed to be a hermetic detector, i.e., it attempts to let no known particles escape the detector undetected. This is because the decay of particles can produce new, neutral, particles that do not interact with any part of the detector. With the hermetic design, imbalance in momentum and energy can be detected, and the production of these non-interacting particles can be inferred.

The current goals of the CMS detector are to provide precise measurements of the properties of the recently discovered Higgs boson, as well as to search for new physics (also referred to as physics beyond the Standard Model) and thereby answer currently open questions in particle physics. What are the properties of QCD (Quantum ChromoDynamics) in extreme conditions? What are the differences between matter and antimatter particles?

It was credited with the discovery of the Higgs boson in 2012, together with the ATLAS (A Toroidal LHC ApparatuS) detector. One of the events displaying a candidate Higgs decay into two electrons and two muons is displayed in figure 1.1.

CMS is designed to be capable to observe particles resulting from proton-proton collisions with a center-of-mass energy of $\sqrt{s} = 14$TeV produced by the Large Hadron Collider (LHC). Currently the LHC provides collisions with a center-of-mass energy of $\sqrt{s} = 13$TeV.

More information about the LHC experiments can be found in the book 'The CERN Large Hadron Collider: Accelerator and Experiments'[4].

1.1 The structure of CMS

1.1.1 Silicon Tracker

The tracker can reconstruct the path of passing muons, electrons, and charged hadrons. Because it is closest to the collisions, the decays of very short-lived particles (e.g. beauty quarks)[5] can be detected.

It makes very precise measurements on the location of particles (with an accuracy up to $10\,\mu$m).

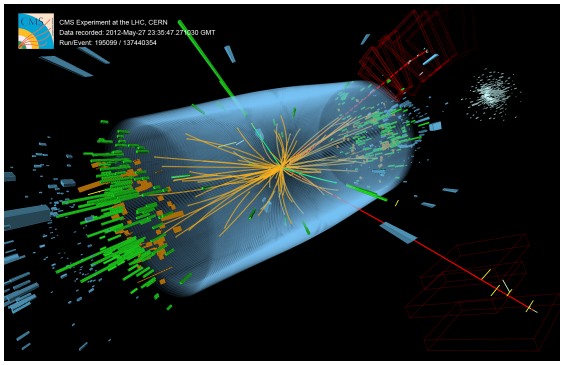


Figure 1.1: Observed candidate decay of Higgs$\rightarrow ZZ^*(ee\mu\mu)$, where the green and red lines emanating from the center are two electrons and two muons, respectively.[1]

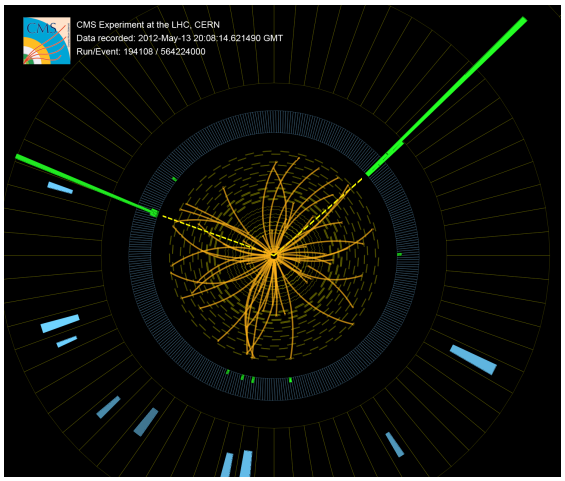


Figure 1.2: Observed candidate decay of Higgs$\rightarrow \gamma\gamma$, the green lines eminating from the center are two photons.[2]

Accompanied by a magnetic field, the momentum of these particles can be determined. Yet this section tries to be as unobstructive to particles as possible. This is accomplished by using a silicon microstrip design, minimizing the volume of material that can obstruct particles.

Being the closest to the proton-proton collisions, this part of the detector is the one that has to endure the most radiation.

1.1.2 Electromagnetic Calorimeter

The Electromagnetic Calorimeter (ECAL) is a set of 75.848 lead tungstate ($PbWO_4$) crystals with a total weight of about 100 tonnes.

It is designed to stop and measure the energy of passing electrons and photons.

These crystals produce light in the form of electromagnetic photon showers (see image 1.3) that are measured by photodetectors (either avalanche photodiodes or vacuum phototriodes).

These crystals are very radiation hard and can reverse their radiation damage when kept in room temperature (during beam time they are cooled to 0.1°C).

1.1.3 Hadronic Calorimeter

The Hadronic Calorimeter (HCAL) is designed to detect hadrons (i.e. particles produced of quarks) and gluons.

This part of the detector is organized in several layers of dense absorbing materials and scintillators, a combination of steel and plastic, where the plastic produces a light pulse when a particle flows through it.

The fact that the HCAL is housed inside the solenoid is one of the biggest differences between the ATLAS and the CMS experiment.

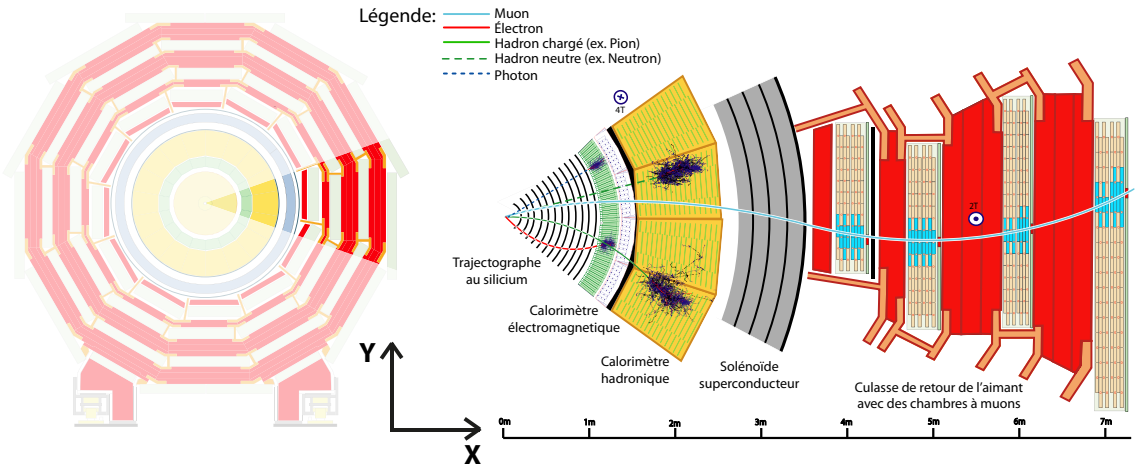


Figure 1.3: A transversal slice through the CMS detector, demonstrating the various sections of the detector and their designed functions.[3]

1.1.4 Superconducting Solenoid

The design of CMS had a high focus on achieving the highest possible magnetic field. To achieve this CMS has been equipped with one large superconducting solenoid, capable of generating a magnetic field of 4 T.

This allows for very precise momentum readings by analyzing the arcs of charged particles. It also provides sufficient return flux outside the solenoid to be used by the muon chambers (about 2 T).

1.1.5 Muon Chambers

Muons (and neutrinos) are the only particles that can pass the previously mentioned sections without losing most (if not all) of their energy.

The loss of energy as a particle travels through matter is described by the Bethe equation (1.1). Because of the characteristics of Muons, this equation states that muons will not release much of their energy as they pass through material.

Combined with the fact that these particles have high mass, this makes them very difficult to stop.

$$-\left\langle \frac{dE}{dx} \right\rangle = \frac{4\pi}{m_e c^2} \cdot \frac{nz^2}{\beta^2} \cdot \left(\frac{e^2}{4\pi\varepsilon_0} \right)^2 \cdot \left[\ln\left(\frac{2m_e c^2 \beta^2}{I \cdot (1-\beta^2)} \right) - \beta^2 \right] \tag{1.1}$$

Since muons can't be easily stopped, the muon chambers rely instead on the strong magnetic field to measure the curvature of these charged particles, and thereby measuring their energy. The muon chambers are composed of three components, the Resistive Plate Chambers (RPC), Drift Tubes (DT), and Cathode Strip Chambers (CSC).

This page is intentionally left almost blank

Chapter 2

The Level-1 Trigger Online Software

2.1 The Level-1 Trigger

The online software is designed to setup, configure, and monitor the electronics responsible for analyzing and filtering data from the CMS experiment as the LHC provides it with sets of proton-proton collisions in the center of the detector at a rate of 40MHz.

The current luminosity of the beam in LHC gives an average of $\tilde{4}$0-50 proton-proton collisions per bunch crossing, resulting in around 2MB[6] of data generated by the sensor electronics.

At a rate of 40MHz this will effectively produce a data stream of 80TB/s. This is too much for any storage system to handle, so a system is needed to filter this data so that only interesting events are retained.

To make the data rate manageable a set of (very fast) algorithms are executed on hardware just outside the detector as events occur to filter out ‘uninteresting‘ data, i.e. physics events that are already known and well-defined. This because collision experiments have been performed for many years now and many observable physics processes in them have already been thoroughly studied.

This filter is called the Level-1 (L1) trigger and will reduce the ‘event rate‘ (i.e. bunch crossing containing proton-proton collisions) from 40MHz to a relative rate of $\tilde{1}$00kHz.

The output data stream of the L1 trigger is then forwarded to the high-level trigger (HLT), which will reduce the event rate from 100kHz to 100Hz. The output data from the HLT is then stored for further analysis by the Worldwide LHC Computing Grid (WLCG). More information about the HLT can be found in the Phase II technical proposal [7].

The calculations of the L1 trigger are performed on FPGA (field-programmable gate array) hardware. These hardware boards follow a common firmware pipeline. The L1 trigger is designed to make a decision every 4 µs, the time it takes for 160 bunch crossings to occur. The trigger will issue a Level-1 Accept (L1A) if this set of bunch crossings is deemed interesting.

The trigger is composed of four levels.

The lowest level, called the Local Triggers or Trigger Primitive Generators (TPG), is a set of hardware boards deployed in the calorimeters as well as the muon chambers, where they are designed to analyze the energy readings and recognize patterns. This level contains the Electromagnetic Calorimeter (ECAL), the Hadron Calorimeter (HCAL), the Cathode Strip Chambers (CSC), the Resistive Plate Chambers (RPC), and the Drift Tube (DT).

The second level combines the information of the Local Triggers and try to make basic reconstructions

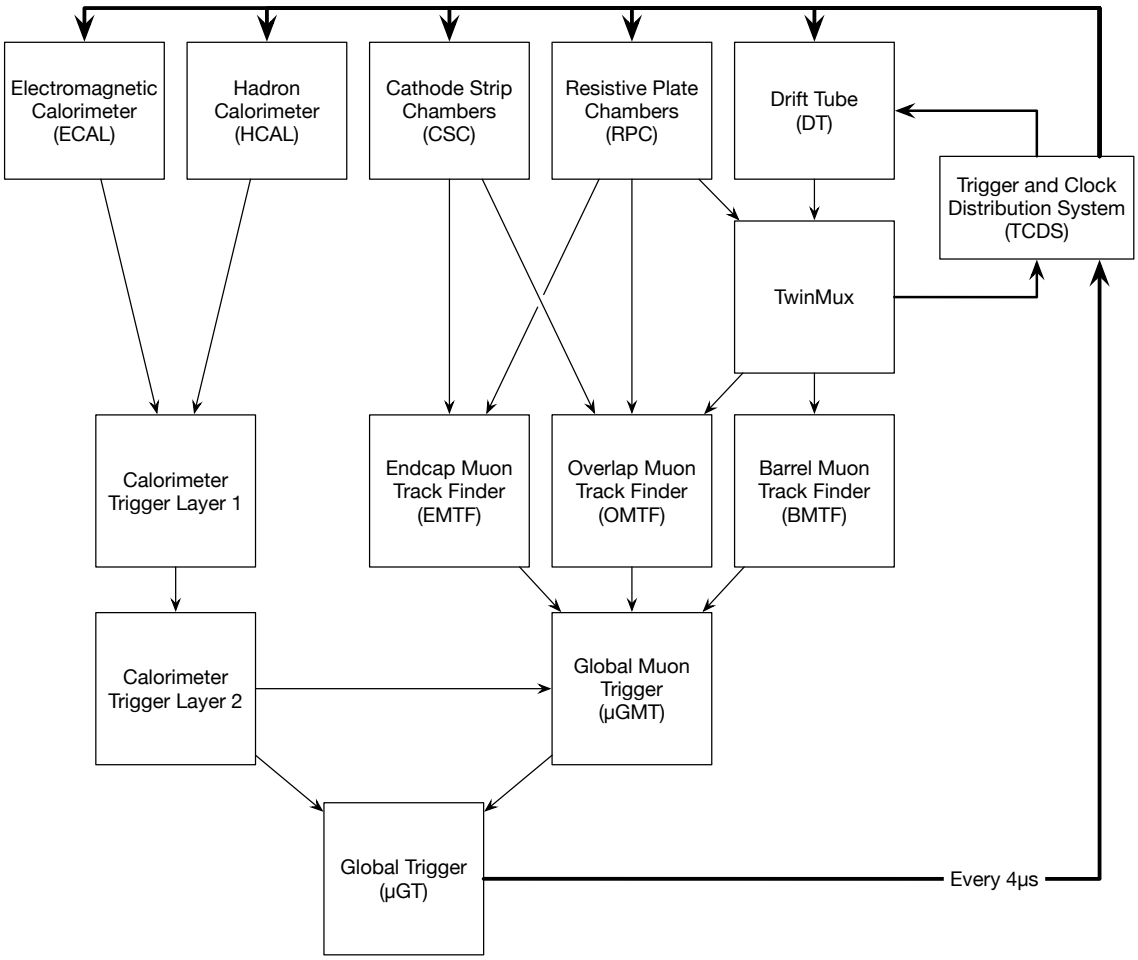


Figure 2.1: Conceptual drawing of the L1 trigger hardware loop

to determine what sort of physics event has happened in that particular region of the experiment. It outputs 'trigger objects', which signify the particle type (e.g. a passing electron or muon) and a rank. The rank is determined by the energy, momentum, and a level of confidence of this measurement. This level contains the TwinMux, the Calorimeter Trigger Layer 1, the Endcap Muon Track Finder (EMTF), the Overlap Muon Track Finder (OMTF), and the Barrel Muon Track Finder (BMTF).

The third level, called the Global Calorimeter and Global Muon Triggers, filter out the highest ranked trigger objects reported by the Regional Triggers.

The highest level, called the Global Trigger, determines whether or not to issue a Level-1 Accept (L1A), and instructs the Timing Trigger and Control System (TTC) to read out the full-precision data stored in the buffers of the FPGA hardware inside the detector. This level contains the Calorimeter Trigger Layer 2, the Global Muon Trigger (μGMT), and the Global Trigger (μGT).

This trigger loop is explained visually in figure 2.1.

Note that, because of the decision deadline of 4 µs, the buffers will contain data of 160 bunch crossings.

Also note that this trigger loop only considers data from calorimeters and muon chambers to limit the data flow through the pipeline.

The Trigger and Clock Distribution System (TCDS) is in charge of distributing a L1A and control signals (e.g. calibration, clock synchronization, test, reset, ...).

2.2 The CMS Experiment Control System

The CMS Experiment Control System (ECS) is a distributed software system that is designed to manage the configuration, testing, and monitoring of all hardware involved in the L1 trigger and DAQ system of the CMS experiment.

One of the components of ECS is the Cross Data Acquisition System, called XDAQ. It is a custom-made data acquisition system specialized for high energy physics. It is developed internally by the CMS group.

XDAQ provides a standardized way to perform high energy physics analyses. It provides developers with a uniform DAQ system. It hides the complexity of data exchange and distributed computing for the developer. It allows subsystems to load its own software modules to perform various tasks. XDAQ also provides an interface engine each subsystem can use to render a web interface.

2.3 The Trigger Supervisor

The Trigger Supervisor is a framework built upon XDAQ which specializes in controlling the various aspects of the L1 trigger and providing libraries for the execution of common tasks performed in most of the subsystems (e.g. configure, test, ...). For example it provides standardized APIs for executing configuration commands and generating monitoring data.

This composes a central system through which the status of all the subsystems of the experiment can be monitored.

The Trigger Supervisor is, as it is built on top of XDAQ, distributed. It is implemented in the form of a tree of ‘cells‘. The Trigger Supervisor has a root cell, called the Central Cell, which has several subcells, each corresponding to a L1-trigger subsystem. Each subcell can contain further subcells. A cell can be a hardware component, or a controller. In the cell tree, all end nodes of the tree are hardware components.

Each cell creates its own web interface dealing with that cell's particular functionality. This way a user can use the web interface to receive a general overview of a system, and traverse the tree to get more specific data and functionality.

2.4 SWATCH

The phase II upgrades of LHC will increase the luminosity of the produced beams. This means there will be more proton-proton collisions per bunch crossing in the experiment, which in turn means there is a bigger amount of data that needs processing.

To support this increase of collision rates, the current hardware at CMS needs to be adapted or replaced to support the higher data rate.

The SWATCH project (SoftWare for Automating conTrol Common Hardware) is an endeavor to standardize the communication with boards and the functions they provide. It attempts to exploit

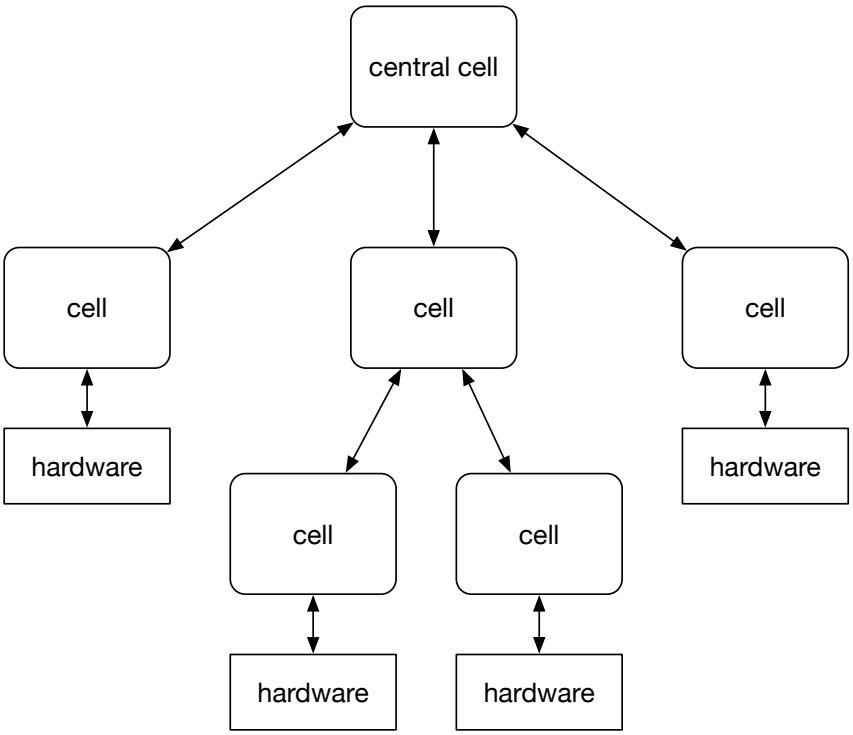


Figure 2.2: An example of the Trigger Supervisor Cell structure

commonalities between hardware components and uses it to provide a common high level API to greatly simplify the development of the online software[8].

Chapter 3

Problems with TS 2.0

The Trigger Supervisor version 2.x had a few problems that caused frustrations with both the operators and the developers of the software.

3.1 Browser compatibility

The first and most visible issue is the slow degradation of support for the interface in modern Web Browsers.

This is due to an effect caused by the movement of major web browser vendors to become ‘evergreen‘, which started around 2011.

3.1.1 Evergreen browsers

An evergreen browser, in essence, is a browser that updates itself without the interaction of the user. This is the formal definition of an ‘evergreen‘ browser, however there are some new philosophies that come with this approach.

First off, a web browser’s version number no longer has a real meaning. To a user a web browser will now be ‘versionless‘, the user will no longer know nor care what browser version is running and will actually assume it is the latest version. Browser vendors have combined auto-updating with a significant speedup of their release cycles. Browsers now tend to update their version once a month, rather than at most once a year (see figure 3.1).

This corresponds to the release early, release often (RERO) software design philosophy. An approach popular in the open-source community and used for the development of the Linux kernel[9].

This in turn engaged web browser vendors to implement new standards much faster and in a much more iterative way than previously possible. Web browser vendors will no longer develop new features as a whole, but rather slowly implement a feature piece by piece. A good example of this is ES6 (sometimes called JavaScript 2015). ES6 is in essence a set of extra JavaScript functionalities and additions to the syntax. Without evergreen browsers, this would have been implemented as one big update, probably in the form of a major release update. However, evergreen browsers implement every ES6 feature bit by bit. This can be tracked with the ES6 compat-table project[10].

Because of these rapid release cycles and automatic updates, evergreen browsers introduce a change in behavior of keeping compatibility with older webpages. If a particular feature would inhibit

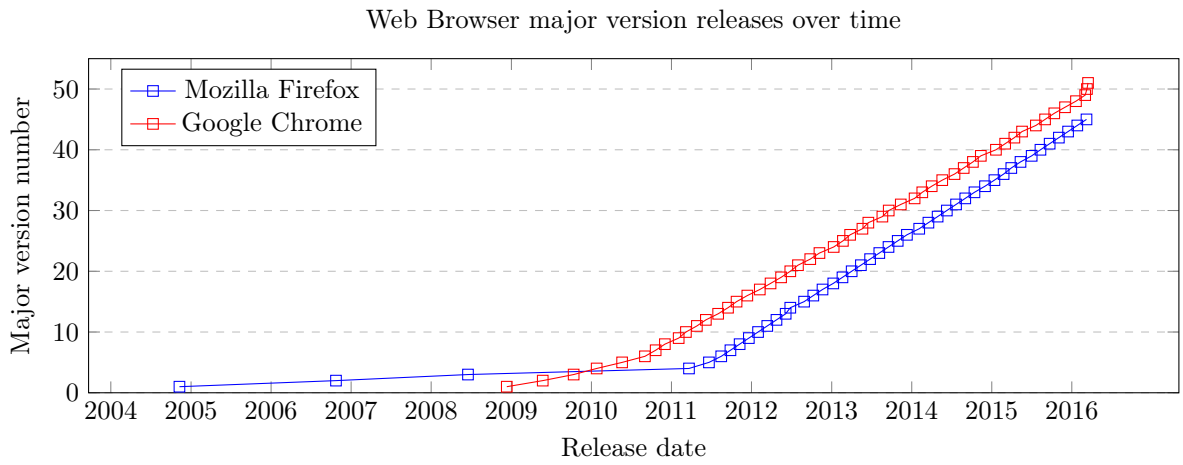


Figure 3.1: Overview of major version releases of Mozilla Firefox and Google Chrome over time

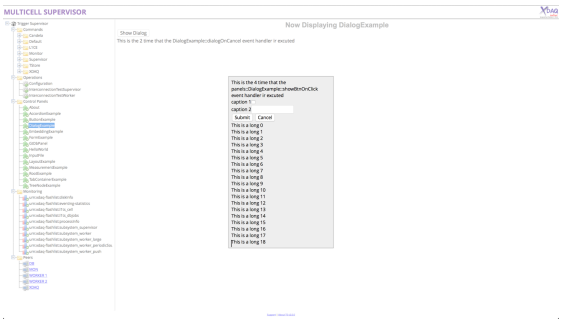


Figure 3.2: The modal dialog in Firefox 38

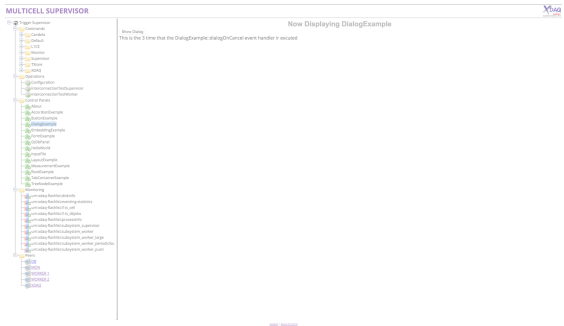


Figure 3.3: The modal dialog in Firefox 44

the development of new features or what is sometimes referred to as 'moving the web forward', it has become acceptable to completely remove that feature. The latest example of this behavior can be found in the rapid implementation, and equally rapid removal, of the /deep/ and ::shadow CSS selectors[11]. This goes against the previous philosophy that a web browser must maintain backwards compatibility with older web pages as much as possible. And this is the main reason the TS 2.0 interface is experiencing problems with modern browsers.

3.1.2 Interface degradation

The age of the TS interface (8 years as of this writing) has reached a point where it uses HTML, JavaScript, and CSS that is being actively removed by browsers. This gives some potentially serious issues when operating the interface.

The most prominent example is the modal dialog feature of the interface. In Mozilla Firefox version 38, the one currently installed in the CMS Control Room, the modal dialog behaves fine. A white overlay is put over the interface and a dialog appears, forcing the user to take a decision. However in the latest version of Firefox, currently 44, the dialog does not appear while the white overlay does appear. This effectively blocks the user from using the interface at all from that point and forces a page reload. Whatever functionality was implemented using the modal dialog is now inaccessible. See figure 3.2 and 3.3.

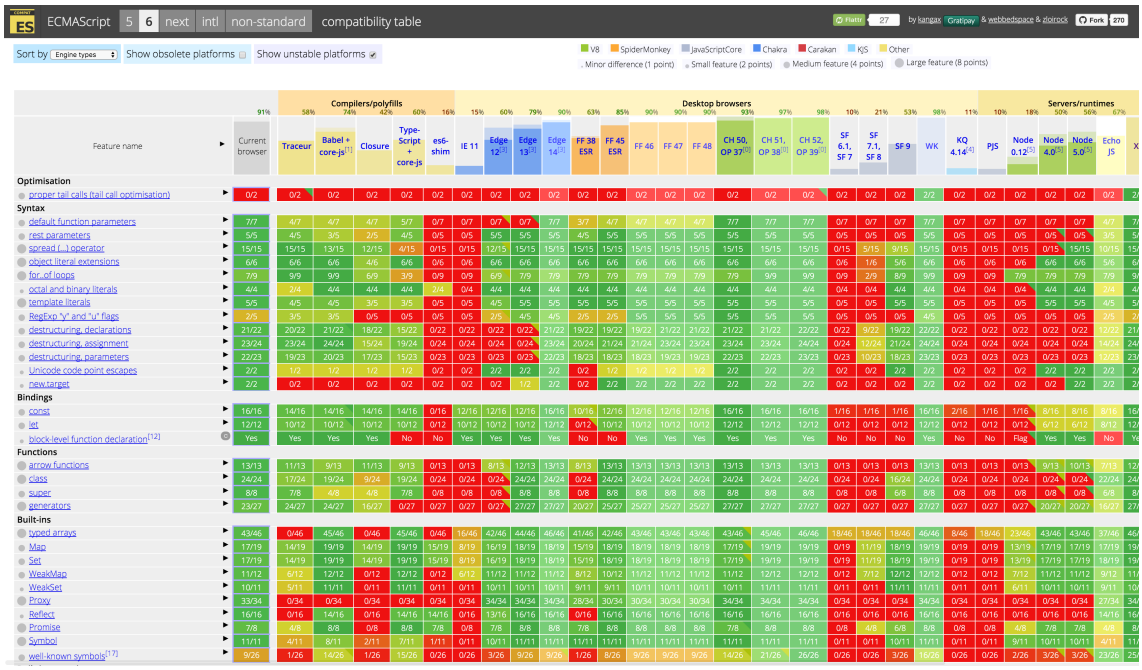


Figure 3.4: The ECMAScript compatibility table project

CERN uses only Extended Support Releases (ESR) of Firefox, and fortunately in this case Firefox ESR deployments at CERN are always about a version behind on the most up-to-date ESR release. This means the interface degradation is currently not breaking functionality yet, however it will in the near future and must be addressed as soon as possible.

3.1.3 Dojo 0.4

The front-end JavaScript framework used to render the interface is called Dojo. It was one of the first JavaScript libraries that successfully attempted to extend the standard html primitives and one of the few frameworks that worked fully client-side.

It was very innovative to pick this framework for the first version of the TS. However, as the version number suggests, it is a beta version. It has some flaws, one of them being a rather huge memory leak issue (discussed in chapter 6.7), another being the ever increasing development time when building interfaces with Dojo 0.4.

3.2 Increasing development time

Dojo 0.4 is around 8 years old now, developed in 2008[12], it cannot be expected to satisfy modern web application requirements.

It misses helper components to set up a layout, something every modern web framework has nowadays, and forces the developer to reinvent the wheel continuously when it comes to interface layouts. This is one of the big reasons interface code tends to become huge and nigh unreadable.

```
1 <!-- A simple title must be coded manually. Prone to typing errors (h2/h4) -->
2 ajax::PlainHtml* title = new ajax::PlainHtml();
```

```
title->getStream() << " <h2 style=\"font-family:arial; color:grey;\"" <<
" align=\"center\">Measurement Example</h4>" << std::endl;
add(title);

<!-- styling must be done manually -->
result_ = new ajax::ResultBox();
result_->setId("subsystem_btnpanel_result_");
result_->set("style","margin:20px; padding:20px; border:2px solid; ");
add(result_);

<!-- some panels need confusing styling to be functional -->
ajax::AccordionContainer* ac = new ajax::AccordionContainer();
ac->set("style","height:80%; width:80%;");
add(ac);
```

It also misses features that cannot be easily compensated. As the requirements for the Phase II upgrade brings increased complexity, it will translate in the framework's need to be able to handle increasing amounts of data reliably. Think for example about large data tables that need filtering and sorting and manipulation while at the same time keeping memory pressure low.

The current framework simply cannot supply this. And all attempts have resulted in an ever increasingly slow interface and complexity in use. For example, an attempt has been made to renew the 'operations' interface. This is an interface that controls a Finite State Machine (FSM) and allows an operator to direct the flow through this FSM and input configuration parameters for each transition. This has been worked on for three months, but was eventually scrapped awaiting the new TS release and it's new ways to develop interfaces.

3.2.1 Maintainability

The fact that simple tasks take much code to implement, combined with the ever increasing complexity required from the interface, results in a maintainability problem. Code becomes unreadable and even small code adjustments take weeks to implement. Larger tasks or new functionality usually are even more challenging to implement.

A recent functionality addition that actually made it into release was the ability to download an arbitrary file from the server. The requirement was to have a download button next to the text area that already contained text of the file that would be downloaded.

It took three different approaches to downloading a file, each implementation more inappropriate than the other. But finally a solution was found, where the file source is just displayed in a new window, allowing for the user to right-click and select 'download source'. Any standard or commonly used way to provide downloading of files ended up being impossible to reliably implement because of the age of the framework TS 2.0 operated on.

This provides another point on why a change was needed.

3.2.2 Large input problem

The Dojo 0.4 framework uses HTTP GET requests with parameters encoded in the url to make requests and post data to the cell. This has some issues one of which recently became a big problem.

First off, HTTP GET, PUT, and DELETE requests should be idempotent. This means that 2 identical requests at different times must produce identical results. Not following this principle creates issues when a proxy server is between the client and the server. A proxy server will always try to cache requests that are supposed to be idempotent. Some HTTP headers exist that allow a developer to instruct a proxy server to not cache a particular request, but it is up to the proxy server implementation to decide if such a request will be honored, and thus cannot be relied on.

Every request currently has such ‘no-cache‘ HTTP headers. And, luckily, no issue with proxy servers has come up yet, however some browser issues are suspected to be linked with this issue.

The second issue is the fact that the parameters of every request are url encoded. This means that parameters are added to the url in the following fashion:

```
http(s)://host:port/path?parameter1=value1&paremeter2=value2
```

The problem with this is that there is a maximum length the url is allowed to be, the exact maximum length depends on both the used browser and server software.

The general consensus is that URLs should be kept under 2KB in size and must not exceed 8KB, as this is where most browsers and server software draw the line. Some panels however, like the operations panel, are designed for very large input variables and far exceeded these limits.

This used to be fine with version 12 of the server software (XDAQ), but in the recently introduced version 13, a hard limit of 8KB has been introduced. This breaks important use cases of panels. It is technically possible to change the Dojo framework’s code regarding request handling. However this might present unforeseen consequences given this is a rather low-level change. Rather it has been decided TS 2.0 will never run under the new XDAQ 13 version.

This page is intentionally left almost blank

Chapter 4

TS 3.0 upgrade requirements

The previous chapter discussed the problems with TS version 2.0.

It is desirable to mitigate all these problems and prepare TS version 3.0 for the future. Several requirements were submitted to the then hypothetical new TS 3.0 components.

4.1 Legacy code compatibility

Currently there are many panels developed in the legacy TS. It is unfeasible to upgrade or rewrite these all in one go. There will be a transitional period where legacy code will have to run concurrently with newer code.

Therefore TS 3.0 must maintain as much code compatibility with TS 2.0 as possible. It is very desirable to have 100% compatibility. A slight area of code incompatibility might have large consequences depending on legacy panel code.

This will put some restraints on the upgrade options as full legacy code compatibility requires the new codebase to be a superset of the old one. However this constraint is mostly applicable only to the server side code, as the client side code is generated by the server and can be significantly modified provided developers keep watchful of any changes.

4.2 Ability to migrate code

As the legacy code will be converted to new code, it would be desirable to make the transition as easy as possible and not to have a too much difference between modern and legacy code as far as keeping existing functionality is concerned.

New functionality will of course result in new code. This requirement therefore only applies to migrating legacy code to keep the functionality as it was.

4.3 Future proof

Web technologies are moving forwards in a fast pace. A lot of new standards have arisen for us to use and it would be wise to use them.

Designing a codebase that uses as much open standards as possible is a good practice. It ensures good support from communities using those standards, and ensures the codebase will maintain compatibility with web browsers for a far longer period than would otherwise be possible.

Provided finalized open standards are used, it would be acceptable to use relatively modern technologies and currently heavily rely on polyfills, libraries designed to emulate a spec not yet implemented, to provide the needed compatibility with currently used software.

4.3.1 Polyfills

A polyfill is a term used in web development. It is a group of JavaScript libraries designed with the very specific use case to implement a future standard, or even just a working draft of a standard, as accurately as possible with today's resources. Some of these also seek to fix a broken implementation of a standard by specific browsers.

Polyfills have gained a lot of popularity the last few years, approximately in tandem with the upcoming of 'evergreen' browsers, as web development now mainly focuses on building on open standards rather than focus on a specific browser or even a specific version of a browser.

Polyfills are generally allowed to introduce as much CPU, memory, and network usage as needed to implement their targeted standard as completely as possible. The main argument for this is that a polyfill is designed to be obsolete after web browsers have caught up and implemented said spec. At this point a polyfill is designed to no longer be activated anymore, mitigating the originally introduced loads.

4.4 Rich functionality

One of the main reasons to renew the codebase of the TS is to be able to fulfill the new modern requirements for the interface.

A lot of new features are needed, like the ability to handle large datasets and the ability to handle more complex analysis use cases.

It would be desirable to have an extendable framework. This way, as time goes on and requirements change, the framework can be adapted and extended to handle new requirements as needed.

4.5 Faster development

The new framework must be much faster to develop on, as currently it takes weeks to implement any change whatsoever.

The current main inhibitors of development time are the lack of features and the amount of unreadable code that the use of the framework causes.

Given the main problems of the slow development time it can be expected that, whatever the new framework looks like, will introduce a major improvement of development time of new interface panels.

4.6 Stability

The new codebase will have sessions at the CMS Control Centre that last for days. This puts important requirements on front-end libraries and frameworks.

Memory leaks are unacceptable. A memory leak results in an unstable interface, which is unacceptable during operations in the Control Centre. This is explained in more detail in chapter 6.7.

4.7 Reduced code footprint

The current code of an interface panel is too large and too messy. It makes the code unreadable and very difficult to maintain.

It would be desirable to have a far smaller code footprint for basic panel functionality. It is a sign of a more powerful framework and will ease the later modifications to panel code.

A smaller amount of required code would translate to the need for a more powerful framework. It must however not introduce functionality that abstracts away it's functionality so much that it would introduce 'black magic' code, i.e. code that works but nobody knows why[13].

4.8 Better maintainability

Functional requirements change regularly. Whatever the new framework looks like must be flexible and open enough to be extended or modified to provide for new functionality as needed. This must either be done in-house or by an extensive and stable developer community associated with the framework.

A 'dead' framework, i.e. one that has no more developer community or ability to be easily modified in-house, like happened with Dojo 0.4 must be avoided.

4.9 Better documentation

The previous requirement of maintainability puts a strong emphasis on documentation. A system can't be properly maintained, nor modified, if the system's workings aren't properly explained.

The framework will need extensive documentation describing the possible ways to program panels and showcase advanced functionalities. Documentation must also exist about the inner workings of the framework.

Documentation must also be easily kept synchronized with the actual state of the code. This would suggest the use of inline documentation, i.e. documentation that resides in the source code.

4.10 i18n

Given the multilingual environment this framework will be used in, it is useful to have a codebase that can adapt to different interface languages.

4.11 Browser compatibility

The TS 3.x front-end libraries must be able to operate on CERN supported browsers. This means it must support any browser installed by default on the currently supported CERN Virtual Machines (VMs). Currently this is Scientific Linux CERN 6 (SLC6) (`http://linux.web.cern.ch/linux/scientific5/`).

This, along with user requests, gives us the following list of browsers that need supporting and their minimum required versions.

- Mozilla Firefox Extended Support Release (ESR) 24-45
- Apple Safari 9
- Google Chrome (latest version)

Notable is the absence of Microsoft Internet Explorer (MS IE) from this list. This is caused by the fact that all production systems use SLC and thus do not run Microsoft Windows.

The Opera and Vivaldi browser shares the same JavaScript and rendering engine as Google Chrome since 2013, when Opera changed to the Blink engine. This also applies to the Vivaldi browser, a popular alternative to the Opera browser. This means that if Google Chrome support is achieved, by definition also Opera/Vivaldi support is achieved.

Chapter 5

TS upgrade options

Now that there is a firm grasp of the current state of the software and its upgrade requirements is achieved, an assessment of the viability of some upgrade options can be performed.

5.1 Server-side interface engine

Old code must run under the new system. Therefore non-backwards compatible modifications to the C++ code are not possible.

The only options are to extend the amount of C++ classes to represent new functionality. There is no restriction in the sense that the old classes have to remain in use. Therefore it is possible to change the way the interface is programmed server-side in function of how extensive the additions to the C++ code are.

5.2 Client-side interface library

At client-side the available options are much more extensive. The old Dojo library can be kept, and any new front-end library can be added provided this new library does not interfere with the functionality of the Dojo library.

Note must be taken however that if any of these new libraries try to dictate the flow of the web interface, it must allow us to alter it to adjust to the page flow that was used by the old codebase.

5.2.1 Upgrading Dojo

Currently the used Dojo version is v0.4. It would seem logical to just upgrade the front-end library to the latest Dojo version, adjust the appropriate C++ classes, and create new ones for the added functionality resulting from the upgrade. There are however a few problems with this approach.

Firstly, starting from Dojo v0.9, there has been a major Dojo API rewrite. This would mean an extensive rewrite of the existing C++ classes is needed, which provides for a very high risk of compatibility problems with legacy panels as they might anticipate some hacky combination of legacy code that would now have dissapeared.

Secondly, it would not have solved any of the problems described in chapter 3, except for the browser compatibility problem.

Running multiple versions of Dojo

Running Dojo v0.4 and v0.9 in parallel is also not possible. There is obvious overlap of code and running these in parallel essentially means entirely refactoring one of the versions, which is undesirable as it is not maintained by us and thus not up to us to perform major modifications to.

However, if possible, it would provide easy migration of legacy code as is would probably mean a panel developer would just have to specify wether to use the new codebase or not.

Dojo 2.0

Currently the latest version of Dojo is v1.10. However the Dojo community is finalising version 2.0. This release, as the version suggests, brings vast modifications to the library.

This version is to be released during spring 2016, which is about a year too late for us to consider. The development started in June 2015 and is expected to be finished before spring 2016. Currently Dojo 2.0 is not stable enough to be considered.

If the software would be upgraded to use Dojo 1.10 now, the new TS would be stuck on an outdated version of Dojo again only months after its release, and no progress will have been made. The community would migrate to Dojo v2.0 while the TS stays behind and face all the current problems all over again.

Browser compatibility

Dojo v1.10 claims the following browser compatibility:

- Firefox 3.6-20
- Safari 5-6
- Chrome 13-26
- IE 8-10
- Opera 10.50-12 (Dojo core only)

This nicely covers and extends the minimum requirements.

5.2.2 jQuery

One of the initial ideas was to replace Dojo with a library like jQuery or Zepto, accompanied by a CSS framework like bootstrap, foundation, semantic-ui, susy, material ui, gumby, Yahoo Pure, or UIKit.

This is a very low-level approach, and for this reason it is actually the safest to ensure compatibility with Dojo 0.4.

However, this approach does not have much more than that to offer. It is not a framework, it is a collection of helper classes and functions. Because it is not a framework, advanced functionality will either be implemented by heavy code duplication or by an in-house developed library.

This will lengthen the initial development time and will also mean longer development times for interface developers.

This solution will not leverage the interface developers like a proper front-end framework will be able to. However, being the most safe option in terms of compatibility, this presents a safe back-up

option if other options end up being incompatible with current needs.

Compatibility with Dojo

Zepto and jQuery functions are fully contained within the '$' namespace. This guarantees the absence of any code overlap with Dojo.

The CSS framework however can still present some problems. Both will try to style common elements like buttons and links, and a decision will have to be made in how to approach this overlap.

Browser compatibility

jQuery combined with the most compatible CSS library (Bootstrap) yields the following browser support:

- Firefox (2 latest versions)
- Safari 7
- Chrome (2 latest versions)
- IE 9

Note the 'last 2 versions' support for Firefox. This could present problems depending on the age of the used Firefox browser. However this list enumerates browsers that have been tested, and since Firefox is generally a 'good citizen' in the world of web browsers it can be assumed this will present no problems. The fact that IE9 is supported strengthens this assumption.

5.2.3 AngularJS

Angular.js is the most popular front-end web application design framework these days. It is very powerful and has an extensive developer community.

It is however not very agnostic about how to design a web app. It makes assumptions, and this could make integrating this with Dojo problematic.

Learning curve

Angular.js is very powerful, but this power is accompanied by a rather steep learning curve[14] that would make it difficult for people who are not full-time web application developers to develop panels for the TS.

Among these problems are confusing terminology (e.g. things called 'constructors' that are not constructors), function parameter based dependency injections that make break minification tools and introduce unnecessary complexity, and an unclear scoping of variables.

Giving that the developers of panels are not programming experts, the learning curve must not be too high, it will greatly increase the required development time to create custom interface panels.

Running Angular concurrently with Dojo

Angular 1.x makes the assumption it has complete control over the layout and the flow of the web application. This will give some issues with the C++ interface building logic when building advanced

interfaces. The C++ code is in control of the application. It constructs the page piece by piece as instructed by the developer and is then to be rendered by the front-end framework.

Angular 2.0

Angular 2 is much more modular, and has a structure quite agnostic and usable to combine with other frameworks.

Implementing Angular 2.0 seems very viable. Unfortunately it is currently still in beta status. And while the basics are stable, advanced functionality is still being debated and developed. This unfortunately limits the use cases for advanced interfaces for the foreseeable future.

Angular 2 Beta could now be implemented while the final release is awaited, but time constraints make this an uncomfortable decision to make.

Browser compatibility

Angular 2 claims the following browser compatibility:

- Firefox (latest development build)
- Safari 7
- Chrome (latest development build)
- IE 9

Firefox (latest development build) looks troubling. Further testing shows that Angular 2 does not work properly in Firefox version <38.

This is very bad, the minimum supported version requirement is v24, 14 versions lower. Most SLC installations have Firefox 38 installed, so this could be workable. However, this is the only option that will not pass the minimum browser support requirements.

5.2.4 Web Components

Web Components[15][16] are additions to the HTML5 standard. They enable a developer to develop custom HTML tags, the idea is to mitigate the ‘div soup‘ problem[17] where the web application’s source code increases exponentially in size as the complexity of the app increases.

This standardizes an approach seen in many modern JavaScript frameworks such as AngularJS (version 2 in particular), Ember.js, Knockout.js, Dojo, and Backbone.js. These all allow a developer to declare new ‘elements‘ in order to make developing a smart web application easier.

However, Web Components are a standardized approach to accomplish this. This means that developers no longer have to worry about major API rewrites like the ones encountered with Dojo.

Furthermore, a vanilla Web Component is guaranteed to be completely compatible with any front-end library. A Web Component is in essence and extra HTML tag and is indistinguishable from a ‘normal‘ HTML tag to a front-end framework.

Web Components consist of the following standards:

Custom Elements This standard allows developers to define their own HTML elements.

HTML Imports This standard provides a way to import an HTML document, much like JavaScript and CSS files are currently imported.

Templates — This standard defines ‘HTML Templates‘ and allows HTML code to be reused as needed.

Shadow DOM — This standard provides a way to have multiple independent HTML DOM trees inside one hierarchy by providing a ‘shadow root‘.

Polymer

Polymer is a relatively new library, built directly on the Web Components standards, developed by Google. It represents the way Google thinks Web Components should be used.

It is very similar to Angular 2 in most respects. For example, they share the same data binding syntax.

The reason Polymer is very useful is that it has the potential to allow us to introduce proper Separation of Concerns (SoC) principles (see chapter 9.3) to the development environment.

Browser compatibility

Web Components are a relatively new set of standards and are currently only supported by Google Chrome.

However, the webcomponents.js project (`https://github.com/webcomponents/webcomponentsjs`) aims to polyfill the Web Components standards.

Using this polyfill, browser support can be extended to the following list:

- Firefox (latest stable build)
- Safari 7
- Chrome (latest stable build)
- IE 10

This list is very similar to Angular 2’s compatibility list. However, testing now concludes that support by Mozilla Firefox goes back all the way to v24, our minimum requirement.

Note that this means that Polymer has better browser support than Angular 2. This is curious, as Polymer uses more recent technologies than Angular 2.

Running Polymer concurrently with Dojo

In a browser with native Web Components support (i.e. the webcomponents.js polyfill is not needed), it is guaranteed to have no conflicts between Dojo and Polymer. This is because Polymer merely adds extras to the Web Components standards and is all contained in the ‘Polymer()‘ JavaScript function.

The webcomponents.js polyfill should also not present conflicts, as most of these polyfills are transparent. The polyfill defines the ‘document.registerElement()‘ function if it doesn’t exist, manually imports HTML Imports if the browser does not support it natively, manually stamps ‘template‘ elements, and defines the ‘element.createShadowRoot()‘ with an approximation to the Shadow DOM spec, called ‘Shady DOM‘, if it doesn’t exist (`https://www.polymer-project.org/1.0/articles/shadydom.html`).

Some quick tests with Firefox v24 confirms that these JavaScript libraries do not present any conflict whatsoever with the Dojo library.

Another possible advantage is that this will probably encounter very little problems with CSS code. As every common HTML element like buttons and links are replaced in Polymer with a more powerful Web Component version (e.g. <paper-button> as a replacement to <input type="button" />).

5.2.5 React.js

React is a JavaScript library, designed by Facebook Inc., that uses a ‘virtual DOM‘ system to abstract away complexity from developers when creating interfaces.

It tries to achieve the same goals as the Web Components standards, however it does not follow these standards but uses custom technologies. For example it uses JSX to define templates rather than the HTML Templates standard.

It has the ability to render server-side and client-side. Server-side rendering is great for web-apps that need good SEO (Search Engine Optimization), however the TS is an internal app. Also this would mean big changes in the server-side code need to be made.

React is an open-source project, but it has a questionable license.

5.3 Chosen upgrade path

5.3.1 Front-end library

Things were very close between Angular 2 and Polymer. They are both the most powerful tools available for front-end interface building today.

Angular 2 inherits its reputation of robustness, stability, and enterprise-level code from its predecessor, Angular 1.x.

Polymer has the advantage of being essentially a small ‘sugaring‘ layer over an established W3C standard. This gives us the advantage of robustness against changes as time advances. Also the backwards compatibility with older browser versions is more extensive with Polymer and the webcomponents.js polyfill than with Angular 2.

The agnostic nature of Polymer will also make future updates easier. As compatibility concerns will be less of an issue.

Seeing that Angular 2 and Polymer try to solve the same problems and share some code syntax, combined with the fact that Polymer is the more standardized of the two and Polymer being the more compatible of the two has led to the conclusion that Polymer is the optimal choice for this project.

React is a notable contender. But the lack of using standards and the custom license are big drawbacks.

5.3.2 Back-end C++ codebase

The current approach of designing interfaces can be adapted to support the Separation of Concerns (SoC) principles (see chapter 9.3).

The C++ code will no longer be in charge of defining interface layout, it will focus and be enhanced for data generation. The interface will be rendered on client-side using Web Components and its templates. Since the main job of XDAQ is Data Acquisition and not interface generation this change of architecture seems suitable.

Existing C++ code will be kept and parts of it can be enhanced to render a web component rather than Dojo code when that Dojo code has stopped working, like the Dojo modal dialog (see figure 3.2 and 3.3 in chapter 3.1.2).

This page is intentionally left almost blank

Chapter 6

TS upgrade roadmap

This chapter will describe the changes that have been made to the interface engine and how compatibility with legacy code is maintained.

6.1 Interface upgrade

6.1.1 The legacy interface structure

The legacy interface engine was programmed entirely in C++ and is developed against using a set of C++ classes that the developer could combine into a tree structure.

Each class corresponds to an element in the Dojo library. These are things like buttons, links, containers, ...

Each class instance in this tree structure has its own string buffer. This string buffer is initially filled with the default HTML and JavaScript code to make the applicable Dojo element work.

Callbacks can be registered and attached to appropriate classes (e.g. an OnClick event callback can be attached to a Button class), this allows the interface to send data back to the server and will in turn allow the developer to change the content of the string buffers.

After such a callback the current interface panel is reloaded and the interface will contain any changes made in any of the string buffers.

Page layout

The main page contains a few div tags that are Dojo ContentPanes, these are used as containers to display the panels, panel menus, and error messages if they come up.

The first ContentPane is ‘tsgui_main_‘, it is placed directly under the body tag and is not used client-side. Rather it is used in the legacy C++ code as a container for the other panels. In ‘tsgui_main_‘, there is ‘tsgui_dummyResult_‘, ‘tsgui_treeBox_‘, and ‘tsgui_content_‘. These are used to display errors, the panel menus, and the interface panels, respectively. This is also shown in figure 6.1.

In the new interface, only the ‘tsgui_content_‘ is kept. This is the Dojo container where all the panels (legacy Polymer) are displayed. The menu and top bar are both handled client-side by a dedicated Polymer element. This is shown in figure 6.2.

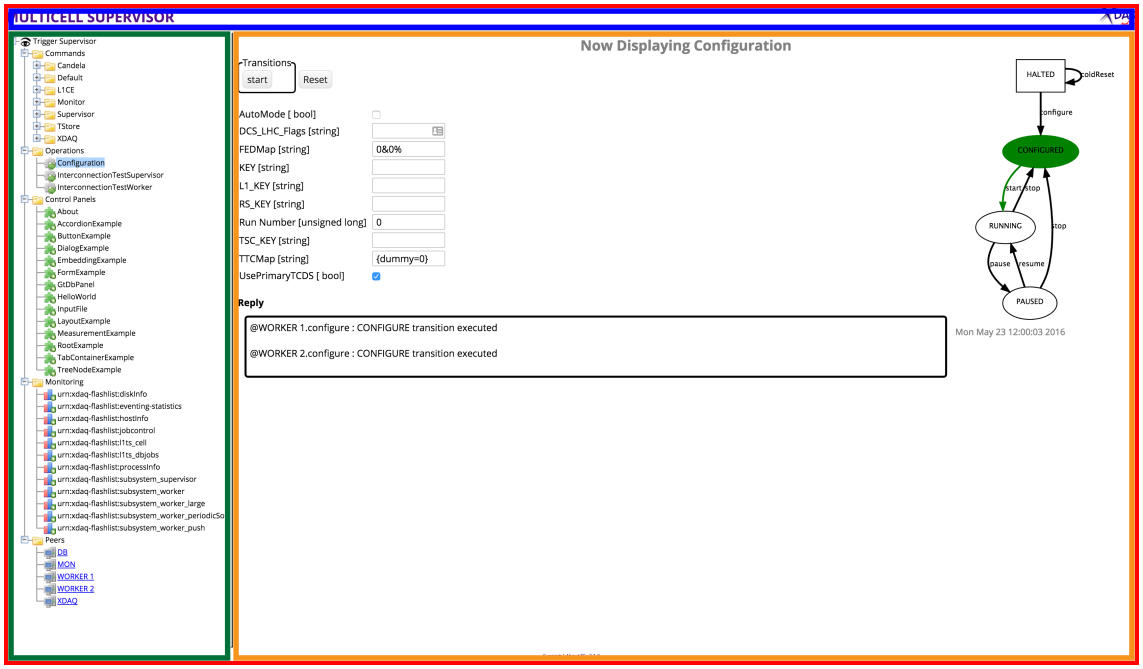


Figure 6.1: Screenshot of TS v2.x with main components highlighted.

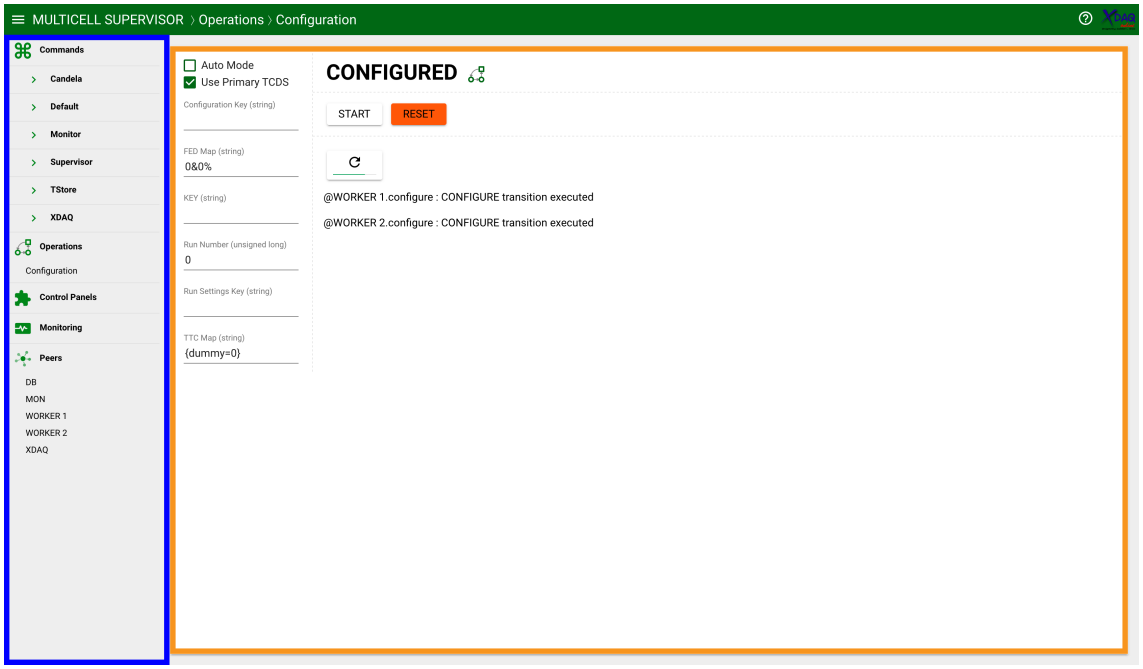


Figure 6.2: Screenshot of TS v3.x with main components highlighted.

Session management

The session is a 48-bit hexadecimal code, and is always encoded in the URL. This way, if the user tries to interact with the server with an invalid session token, the server can simply respond with a JavaScript payload redirecting them to a URL containing a valid session ID.

The URL always looks like this

```
http(s)://host:port/Default?_sesionid_=0x000000000000
```

Any state, e.g. the currently loaded panel, is kept server-side in the string buffers.

6.1.2 The new page structure

The new interface will have the same general structure as the legacy interface, though updated with the material design look and feel (see chapter 6.2.1), and some improvements (like a breadcrumb trail).

6.1.3 Emulation of the legacy structure in the new page

The new main page will keep the ‘tsgui_content_‘ Dojo ContentPane. This way legacy Dojo panels can still be served, and since the ContentPane is just a plain ‘div‘ tag when not using Dojo it won’t inhibit any new code from functioning properly.

6.2 New session management

The session ID will move out of the URL and will be moved into the response headers of the server, this header will only be sent if a session change happened.

This approach has a few advantages:

- The session can now no longer be accidentally shared between users, since it is no longer contained in the URL.
- A session renewal does not require a full page reload. The panel will need to be reset, as the user received a new session, but this will now be much faster as it doesn’t require a full reload.
- In the old session system the user was navigated to the default page on a session renewal, and did not preserve the user’s navigation through the cell interface like the new session system does. In the new system the interface detects the presence of a new session id in any response of the server, and executes appropriate code to handle a session renewal.

6.2.1 Material Design

Material Design (previously called Quantum Paper) is a design language developed by Google[18], released in 2014. It aims to return to the design principles used in printing, and extend it with things that are normally not possible in real printing (e.g. motion, responsive layouts, . . .).

At the center of material design is paper, every component of the design spec treats a user interface element (e.g. containers, buttons, dropdowns, ...) as a it was being cut out and pasted together with paper. The reason for this is that it is easier for a user to think with physical objects,

Google uses it to bring back consistency throughout its product line and across all types of devices. Material design is deployed on watches, phones, tablets, laptops, and televisions. Android Marshmallow (v6.x) has fully migrated to material design, and the vast majority of apps in the Google Play store have adopted it.

The full Material Design spec can be found on the Google design webpage[19].

6.3 Handling large input

The legacy (i.e. Dojo) panels sent data back to the server using URL encoding (more info in chapter 3.2.2).

The legacy panels can't be adjusted without risking unforeseen consequences. The new (i.e. Polymer) panels however will all send data back to the server properly, using HTTP POST requests that contain any parameters in the POST body.

This is the way the HTTP specification designed to send data back to a server, and thus it solves the problems encountered with sending data with the legacy code. With the new approach it is now theoretically possible to send multi-gigabyte sized data.

6.4 Additions to the page builder classes

The C++ classes in the legacy system each represent an HTML element. This is manageable because in the set of elements in the legacy system is fixed. Since in the new system every interface developer will now have the possibility to extend the set of elements it would make sense to make a more abstract class to handle these new set of elements.

The ajax::PolymerElement class was added and is used like follows:

```
1 ajax::PolymerElement* myElement = new ajax::PolymerElement("my-element");
2 myElement->set("some-property", "someValue");
3 add(myElement);
```

Furthermore, the PlainHtml class has been extended with some shorthand functions to make it easier to use. This because it is a frequently used class. instead of doing:

```
1 ajax::PlainHtml* br53 = new ajax::PlainHtml?();
2 br53->getStream() << " <p>some html code<p>";
3 add(br53);
```

it is now possible to do this:

```
1 add(new ajax::PlainHtml(" <p>some html code<p>"));
```

This greatly enhances readability of pages containing a lot of arbitrary html code like '
'

The AjaXell code has also been modified to support HTML5 features like boolean attributes (i.e. attributes with no value).

6.5 Upgraded event system

In the legacy codebase, events are attached to C++ classes declared in a panel.

```
ajax::Button* button = new ajax::Button();
button->setId("subsystem_btnpanel_button_");
button->setCaption("click me!");
this->setEvent(button,ajax::Eventable::OnClick,result_,this,&ButtonExample::onClick);
add(button);
```

However, this system has been extended to allow an event to be attached to the panel itself, rather than a class instance defined in the panel code.

```
setEvent("submit", ajax::Eventable::OnClick, this, &FormExample::submit);
add(new ajax::PolymerElement("form-example"));
```

This is necessary because the server will no longer render every element in the interface. An element (e.g. a button) can be rendered client-side, so

6.6 New JSON library

The primary job of the server-side code will no longer be interface generation, but data generation.

In the legacy codebase there has been no easy way to generate XML or JSON data. The only viable way to construct these were to construct them manually or to use BOOST Property Trees, which require extensive amounts of code.

The TS will incorporate the JsonCpp library, a lightweight C++ library that makes the generation and parson of JSON very easy.

More information about this can be found in chapter 9.1.3.

6.7 Memory-leak problem

An interface panel can be used for extensive amounts of time, this time can be expressed in days. Therefore any memory leaks are unacceptable.

Unfortunately it is rather easy to create memory leaks in JavaScript. JavaScript uses a Reference-counting garbage collection system[20]. Such a garbage collector cannot recognize circular references, and JavaScript closures add another memory leak pattern to watch out for.

6.7.1 Memory-leak patterns

Circular references

A circular reference is formed when two or more objects reference each other in such a way a closed circle can be drawn.

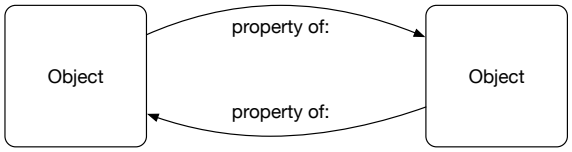


Figure 6.3: A circular reference

In a reference counting garbage collector like JavaScript these kind of structures present problems because there is no way the reference count of any of the object in a circular reference can reach zero, and thus will never be garbage collected.

The following code will create a circular reference.

```
<html>
  <body>
    <div>an HTML element</div>
    <script>
      var div=document.querySelector("div");
      div.someproperty = div;
    </script>
  </body>
</html>
```

This piece of code will however not appear very often. There is not a real use case where this would appear, and the memory leak is rather obvious.

JavaScript closures

A feature of JavaScript is that functions can contain other functions. The inner function will inherit variables from the parent function.

This inheritance of variables by inner functions is called closure. This is a code example demonstrating closure:

```
window.onload = function() {
  var test = 5;
  function innerfn() {
    // will display 5
    alert(test);
  }
  innerfn();
}
```

Closures are the cause of most memory leaks in JavaScript. Consider the following use case:

An element is created, and a callback is attached to it to react to the click event.

```
var newelement = document.createElement('p');
newelement.textContent = 'click me';
newelement.onclick = function() {
  alert("you clicked me");
}
```

```
6 document.body.appendChild(newelement);
```

This simple code example contains a memory leak. the function attached to the onclick event inherits the ‘newelement‘ variable, and thus has created a circular reference.

Avoiding memory leaks

Some simple patterns exist to avoid circular references.

Firstly, in a parent function one can set one of the variables causing a circular reference to null, thereby breaking the circle.

```
1 var newelement = document.createElement('p');
2 newelement.textContent = 'click me';
3 newelement.onclick = function() {
4   alert("you clicked me");
5 }
6 document.body.appendChild(newelement);
7 newelement = null;
```

Another approach is to insert another closure

```
1 var inner2 = function() {
2   alert("you clicked me");
3 }
4 (function inner() {
5   var newelement = document.createElement('p');
6   newelement.textContent = 'click me';
7   newelement.onclick = inner2;
8   document.body.appendChild(newelement);
9 })();
```

Lastly, one can avoid closure altogether.

```
1  function callbackfn = function() {
2    alert("you clicked me");
3  }
4  var function makeElement = function() {
5    var newelement = document.createElement('p');
6    newelement.textContent = 'click me';
7    newelement.onclick = callbackfn;
8    document.body.appendChild(newelement);
9  }
10 makeElement();
```

6.7.2 Memory leaks in Dojo

Unfortunately, Dojo 0.4 or the implementation used here seems to contain a lot of circular references. Memory usage goes up linearly with the amount of panels used in a browser session.

To test this, a panel will be reloaded (by clicking in the menu) every second. This will be done for

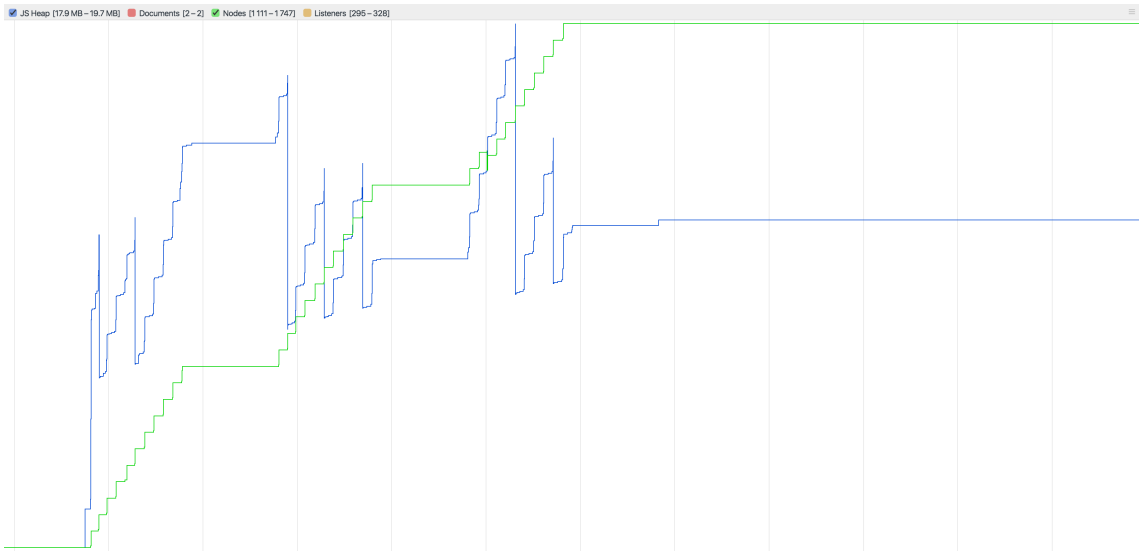


Figure 6.4: TS 2.x interface memory usage test

a period of ten seconds, following a 10 second idle period to allow for any garbage collection to trigger. This cycle will be executed for sixty seconds, and will be followed by a final sixty second idle period to make sure any garbage collection has occurred.

The results for this test for TS v2.1.0 is displayed in figure 6.4. Notice that the amount of nodes does not go down. This is because of the circular references described earlier. The garbage collector will never remove these nodes from memory, because the reference count of these nodes will never reach zero.

As a result, the memory consumption will grow arbitrarily as panels are used.

In the TS 2.x interface this was not noticed because of the very frequent full page reloads. However, with the new graceful session renewal that does not require a page reload, these memory leaks in dojo panels will pile up until the user manually refreshes the browser.

Figure 6.5 shows the result of the same test in the TS 3.x interface, with the same legacy panel used to test the TS 2.x.

Despite the rewritten main interface, the memory leak still occurs with legacy panels. This is because the cause of the memory leak resides inside the Dojo library itself.

A solution to this problem is not easily apparent. It would require changes to the legacy codebase, which is always a risky thing to do. It is decided that for this reason, any legacy panel that features automatic refreshes (e.g. the operations panel) has a high priority to be replaced with a polymer alternative.

6.7.3 Memory leaks in Polymer

The same test is done again in the TS 3.x interface, but with the panel being upgraded to a Polymer equivalent. The results are shown in figure 6.6.

Note that this time, the garbage collector is able to remove unused DOM nodes from memory, followed by the memory usage returning to its original value immediately after the test.

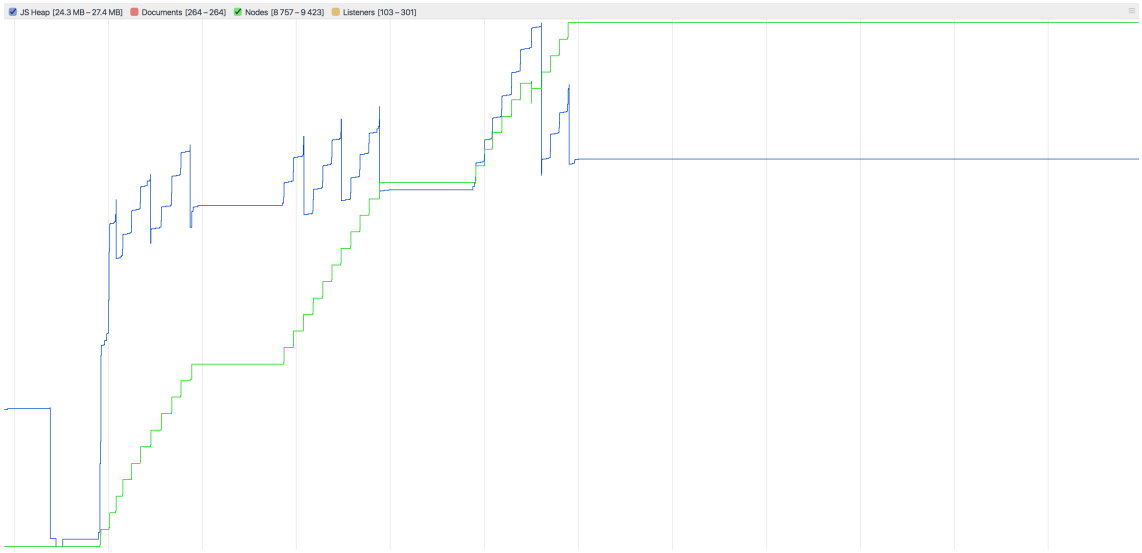


Figure 6.5: TS 3.x interface memory usage test with a legacy panel

Figure 6.6: TS 3.x interface memory usage test with a Polymer panel

This page is intentionally left almost blank

Chapter 7

Development process

Given the size of the TS and the amount of stakeholders in the proposed changes, it is important to have a decent planning.

The main objectives are productivity, and making sure developers deliver what is needed most at a particular moment.

It also would be nice to have a sensible feedback system to allow the stakeholders to have an influence in the evolution of the TS changes, as they will be the people who are going to use it.

A modified version of the Scrum method has been used as the development process. The modifications are focused on providing compatibility with the other development processes in use by other developers. Also it has been modified to account for the limited amount of people working on one task.

7.1 Scrum

Scrum is a relatively new way of developing software projects, although it has also been used to execute projects not related to development (e.g. construction projects).

It focuses on delivering functional requirements prioritized on the value it adds to the project as a whole.

It implements a very strict and repetitive development cycle, usually with a period of 1 or more weeks. This is called a Sprint. During the beginning of a Sprint, a set of functional requirements are chosen as a goal, and must be achieved by the end of the Sprint.

An important distinction to make is that the developers themselves drive this process. There is no separate person who makes the planning and the set of requirements for a Sprint on their behalf. This is where Scrum gets efficient, because the developers after all know best what is most important and feasible to achieve in one full Sprint.

At the end of a Sprint a set of functional requirements must be met. This means that particular set of functionality in the project must work in a sense that the end user can use it. This must be proven by a working demo to the stakeholders of the project, all of them.

This is also a very important part of Scrum. By demanding the functionality must work to such an extend that it would be useful for deployment means there is far less opportunity for hidden errors during actual deployment. The working demos to all of the stakeholders also provide feedback to developers at early stages, unlike other systems where stakeholders only get to see the product all the way at the end of the product development and notice the developers and stakeholders had

some different ideas about functionality.

For more info about Scrum can be found in the book written by one of its inventors, Jeff Sutherland[21].

7.1.1 Kanban

Tasks are divided into distinctive and sequential states. Each task must flow through each state. A change in the task results in that task being reset to the initial state.

The following states were chosen for a task:

Backlog This is a list of all the tasks that need to be done. They are not part of a Sprint, but are the list of candidate tasks for a Sprint.

To do Tasks get moved from the backlog to ‘To do‘ when they are selected to be part of the currently starting Sprint. This list represents the set of tasks that need to be competed (i.e. be in the ‘Done‘ state) by the end of the Sprint.

In progress When someone is working on a particular task, it is moved from ‘To do‘ to ‘In progress‘.

In review A task get put into ‘In review‘ when it is considered ready for use. At this stage another developer double checks the new functionality. The main objective is detecting missing features or a misunderstood implementation of it.

Testing At this point the code of a task is pushed to the SVN repository. The relevant code is then recompiled and tested by a few experts (i.e. people who will be using this panel).

Done After testing is complete, a task is considered ‘Done‘ and awaits a new software release to be put in ‘In production‘.

In production Once a new release of the TS is pushed to production systems, the appropriate tasks are moved to ‘In production‘. A task in this state can be deleted from the point it can be reasonably assumed the relevant features are stable.

Trello (`https://trello.com/`) has been the tool of choice to implement this Kanban board (see figure 7.1).

7.1.2 Workflow

The Scrum process has been modified to account for the tiny number of developers.

Every week a list of functional requirements is made, preferably this does not encompass any technical goals and thus only contains goals towards end-user functionality. These are formed into tasks and get put into the backlog.

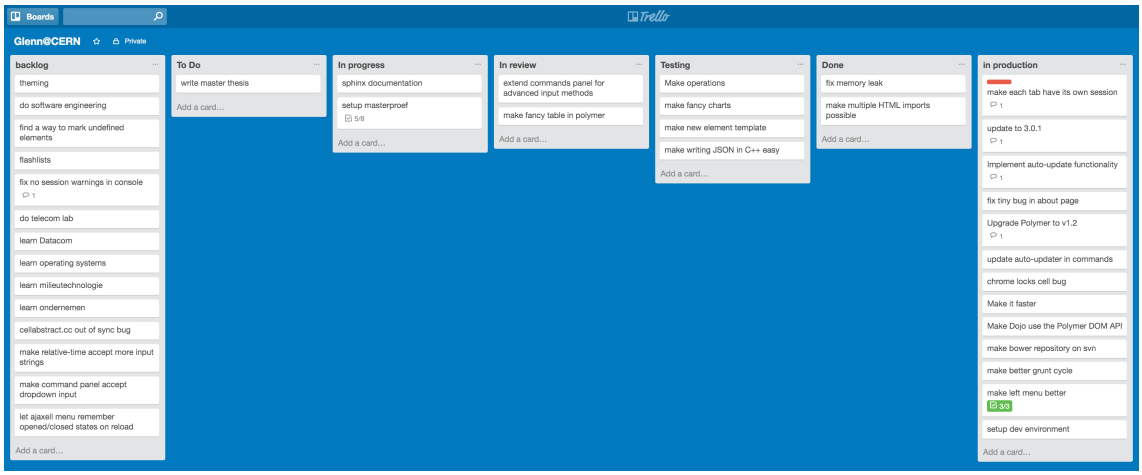


Figure 7.1: Screenshot of the Trello Kanban board used during development

This list is then sorted according to urgency and importance (urgency takes precedence over importance). After sorting, the backlog items are considered to be put into ‘To do‘ status up until a point the Sprint contains enough tasks.

After the tasks have been done, they are in the ‘In review‘ stage. Where they will be either reviewed internally or reviewed during the weekly or monthly meetings depending on the importance of the feature.

7.2 Version Control

Apache Subversion (SVN) is used to implement version control with the online software source code.

It is accompanied by a web based ticketing system based on Trac (`https://trac.edgewall.org/`).

The repository structure follows all common best practices. The ‘trunk‘ folder contains strictly working and tested code and is used to perform the nightly builds. It has a ‘branches‘ folder containing any pending bug fixes or added features. Branches follow the ‘username_foldername_ticket#‘ naming convention. The repository also has a ‘tags‘ folder, containing working copies of the source code that are associated with a specific version (e.g. 2.0.1).

The versioning system uses three numbers to signify major, minor, and patch changes, respectively.

This repository can be found at `https://svnweb.cern.ch/trac/cactus`

This page is intentionally left almost blank

Chapter 8

Polymer

Polymer allows a developer to declare new web Components. This is used to generate the custom web interfaces.

This chapter describes how Polymer is used and what developers can do with it.

8.1 From C++ to Polymer

As stated in chapter 9.3, C++ should only be used for data generation.

However, to keep compatibility with legacy code, the string buffer system is still used. Therefore the interface must be initiated like so: (this example is taken from the Flexbox layout example in the Subsystem Supervisor)

```
1  #include "subsystem/supervisor/panels/FlexboxLayout.h"
2  #include "ajax/PolymerElement.h"
3
4  using namespace subsystempanels;
5  FlexboxLayout::FlexboxLayout( tsframework::CellAbstractContext* context,
6                                log4cplus::Logger& logger)
7  :tsframework::CellPanel(context,logger) {
8    logger_ = log4cplus::Logger::getInstance(logger.getName() +".FlexboxLayout");
9  }
10
11 void FlexboxLayout::layout(cgicc::Cgicc& cgi) {
12   remove();
13   add(new ajax::PolymerElement("flexbox-layout"));
14 }
```

There is still a tiny piece of C++ code that renders the initial HTML code, at line 13. It merely specifies the name of the Web Component that renders the panel interface for the Flexbox layout example panel.

8.2 From Polymer to C++

The previous example did not contain any data generation.

In order to add support for a callback, it must be registered like so: (this example is taken from the Form example panel in the Subsystem Supervisor)

```
#include "subsystem/supervisor/panels/FormExample.h"
#include "ajax/PolymerElement.h"
#include "ajax/toolbox.h"
#include "json/json.h"

using namespace subsystempanels;
FormExample::FormExample( tsframework::CellAbstractContext* context,
                          log4cplus::Logger& logger)
:tsframework::CellPanel(context,logger) {
  logger_ = log4cplus::Logger::getInstance(logger.getName() +".FormExample");
}
FormExample::~FormExample() {
  remove();
}

void FormExample::layout(cgicc::Cgicc& cgi) {
  remove();
  setEvent("submit", ajax::Eventable::OnClick, this, &FormExample::submit);
  add(new ajax::PolymerElement("form-example"));
}

void FormExample::submit(cgicc::Cgicc& cgi,std::ostream& out) {
  std::map<std::string,std::string> values(
    ajax::toolbox::getSubmittedValues(cgi)
  );

  Json::Value root(Json::arrayValue);

  for(
    std::map<std::string,std::string>::iterator i(values.begin());
    i != values.end(); ++i
  ) {
    Json::Value someresult;
    someresult["name"] = i->first;
    someresult["value"] = i->second;
    root.append(someresult);
  }

  out << root;
}
```

Now a callback ‘submit‘ is registered, which will execute the FormExample::submit function and return whatever output it generated.

These callbacks can be implemented in Polymer using the ‘ts-ajax‘ or the ‘auto-update‘ element in the ‘common-elements‘ package.

```
<dom-module id="form-example" >
  <template>

```

```
4     <ts-ajax id="ajax"
5              callback="submit"
6              data="{{response}}"
7              parameters='["sometextinput"]'
8              sometextinput={{text1}}></ts-ajax>
9     <paper-input label="some text input"
10                 value="{{text1}}" ></paper-input>
11
12    <paper-fab icon="send" title="send" on-click="submit" ></paper-fab>
13
14    <h1>server response:</h1>
15
16    <template is="dom-repeat" items="[[response]]" as="item" >
17      <p>[[item.name]] = [[item.value]]</p>
18    </template>
19
20  </template>
21  <script>
22    Polymer({
23        is: 'form-example',
24
25        properties: {
26          response: {
27            type: String,
28            value: undefined
29          }
30        },
31
32        submit: function() {
33          this.$.ajax.generateRequest();
34        }
35    });
36  </script>
37 </dom-module>
```

8.3 Polymer features

Polymer is bundled with a lot of useful functionality out of the box. The most used ones in this project are described here.

A more detailed explanation can be found at the Polymer docs[22]

8.3.1 Properties

Polymer elements can contain custom properties defined by the developer. A property can be an object, a date, a string, a number, an array, a boolean, or a function. They can be configured to be read-only, to spawn events and/or execute functions when their value changes.

Properties are useful to display to, or get data from, the user.

8.3.2 Data binding

Data binding is used to connect properties between web components or to bind a property to a control like an input box or a button value.

It is also useful to automatically fetch data from the server and put it in the interface without much effort. An example of this can be found in chapter 8.2.

8.3.3 Lifecycle callbacks

A web component can react to lifecycle events. This allows a developer to write code to be executed when a new instance of the web component is created, or when it is placed inside the DOM tree, or removed from it.

A useful example is the ‘auto-update‘ element, which declares an interval on the ‘attached‘ event, and removes it on the ‘detached‘ event.

8.3.4 Styling

AjaXell provides developers with a theme. This is provided in the form of a web component called ‘reset-css‘. It makes other web components conform to the AjaXell them (e.g. button size, fonts, colors, . . .).

Also Polymer gives developers the possibility to use CSS functionality that is not yet implemented in all browsers, but which are very useful for web components.

One of them is CSS variables and mixins, which allows a developer to define variables in CSS. Notable is that these variables are inherited just like normal properties. This is how the AjaXell theme file defines its theme colors.

```
:root {
  --primary-color: #00671a;
  --warning-color: yellow;
  --text-color: black;
  --fancy-gradient: {
    background: linear-gradient(to bottom,
                                #1e5799 0%,
                                #2989d8 50%,
                                #207cca 51%,
                                #7db9e8 100%);
  }
}
paper-button {
  color:var(--text-color);
  --paper-button-ink-color:var(--primary-color);
}
paper-button.warning {
  --paper-button-ink-color:var(--warning-color);
}
div[fancy] {
  @apply(--fancy-gradient);
}
```

A Polymer element can include this theme by including it in its template.

```
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html" >
<dom-module id="my-element" >
  <template>
    <style include="reset-css" ></style>
    <paper-button>I am themed!</paper-button>
  </template>
  <script>Polymer({is: 'my-element'});</script>
</dom-module>
```

8.3.5 Events

An element can define define events. It can also react to any event spawned by elements in its template using the following syntax:

```
<dom-module id="my-element" >
  <template>
    <!-- paper-button fires 'click' event -->
    <paper-button on-click="clicked" >Click me!</paper-button>
  </template>
  <script>
    Polymer({
      is: 'my-element',
      clicked: function(e, detail) {
        this.fire('kick', {kicked: true});
      }
    });
  </script>
</dom-module>
<my-element></my-element>
<script>
  document.querySelector('my-element').addEventListener('kick', function (e) {
    if (e.detail.kicked) {
      console.log("kick event fired");
    };
  });
</script>
```

8.3.6 Behaviors

Behaviors in Polymer is a way to share JavaScript code among multiple elements. A useful example of this is the ‘throws-toast‘ behavior in common-elements, which allows an element to show notifications to the user.

The throws-toast.html behavior definition looks like this:

```
<script>
throwsToast = {
  /**
    * Throw a message to the central toaster.
```

```
    * The 'Toast' object accepts the following properties:<br>
    * type (String): Can be 'info', 'warning', or 'error'.<br>
    * message (String | HTMLElement): The actual message to show.<br>
    * options (Array of Strings) (optional): This will force the user
    * to stop what they're doing and choose one of the supplied
    * options. The chosen option is returned.
    *
    * @param {Toast} toast Can be 'info', 'warning', or 'error'.
    * @param {function} callback The function to invoke when one of
    * the options has been clicked.
    * Takes the option string as argument.
    * @return {Void | String}
    */
  throwToast: function(toast) {
    if (typeof toaster === "undefined") {
      console.error(this, "toaster is undefined");
    } else {
      toaster.throwToast(toast, this);
    }
  }
};
</script>
```

An element can implement it like this:

```
<link rel="import" href="/ts/ajaxell/html/elements/ag-toaster/throws-toast.html" >
<dom-module id="notifications-demo" >
  <template>
    <paper-button warning on-click="showWarning" >show warning</paper-button>
  </template>
  <script>
    Polymer({
      is: 'notifications-demo',
      behaviors: [throwsToast],
      showWarning: function showWarning() {
        this.throwToast({
            'type': 'warning',
            'message': 'this is a warning at t=' + new Date().getTime(),
            'callback': function callback(response) {
                console.log("callback successfull: ", response);
            }
        });
      },
    });
  </script>
</dom-module>
```

Chapter 9

The renewed panel SDK

The new front-end and back-end codebase provide an interface developer with a whole new set of tools.

9.1 Packages available to panel developers

The set of pre-made tools available to developers has vastly expanded.

Not only does this mean a more rich set of tools are developed internally, developers can now also find and include tools they find on the web.

This will provide a way to get the framework capabilities to evolve as time goes on.

9.1.1 Bower Components

All libraries, web components, sources, ...that are not developed in-house are housed in the bower-components package. The name is derived from the package manager used to pull in these resources, called 'bower' (`http://bower.io/`).

The bower-components package contains a 'bower.js' file that specifies the dependencies to be pulled from the web. Then the dependencies are compressed into a tarball. This is done to make sure the package versions do not change unintentional, and new package versions can be tested in a controlled manner.

Currently, the following elements are included in bower-components:

Polymer (`https://www.polymer-project.org/`)

iron-elements (`https://elements.polymer-project.org/browse?package=iron-elements`) The iron-elements are a set of web components aiming to provide a basic set of tools and enhancements to standard elements, for example to provide them with data-binding capabilities.

These elements do not make assumptions about the used layout or styling, and are expected to maintain a spartan view, if they render a view at all.

Iron-elements aim to extend basic html elements (e.g. <iron-input> to extend <input>), provide façade elements for javascript functionality (e.g.

<iron-ajax> to easily make AJAX requests), or provide new functionality that would be considered basic functionality (e.g. <iron-icon> to display an icon).

paper-elements (https://elements.polymer-project.org/browse?package=paper-elements) Paper-elements is a set of elements that focus on bringing Material Design[19] to web components.

Paper-elements aims to extend iron-elements with material design (e.g. <iron-input> becomes <paper-input>), and introduce new elements that are unique to material design (e.g. <paper-toast>)

gold-elements (https://elements.polymer-project.org/browse?package=gold-elements) Gold elements are input elements for specific use cases (e.g. email, phone numbers, credit card numbers, ...).

They all extend the 'paper-input' element and provide specific validation and formatting functionality.

neon-elements (https://elements.polymer-project.org/browse?package=neon-elements) neon-elements are a set of Web Components designed to be façades for the JavaScript animation API to make them available by purely writing HTML.

These elements do not use CSS Transitions, CSS Animations, or SVG, rather they use the new Web Animations API (https://www.w3.org/TR/web-animations/).

These are among the most advanced Web Components in the packages available to panel developers. More info about their usage is provided here: https://youtu.be/-tX0e29GQa4.

platinum-elements (https://elements.polymer-project.org/browse?package=platinum-elements) Platinum-elements are a set of Web Components focused on providing a façade for web-app capabilities like Service Workers, server push, and bluetooth connectivity.

jQuery (https://jquery.com/)

moment.js (http://momentjs.com/)
JavaScript library for parsing, validating, manipulating, and formatting dates.

page.js (https://visionmedia.github.io/page.js/)
Micro client-side JavaScript router inspired by the Express router.

spectrum.js (https://bgrins.github.io/spectrum/)
Spectrum is a JavaScript colorpicker plugin using the jQuery framework.

juicy-jsoneditor (https://github.com/Juicy/juicy-jsoneditor/)

cytoscape (http://cytoscape.org/)

paper-datatable (https://github.com/David-Mulder/paper-datatable)

vaadin-core-elements (https://vaadin.com/elements)
Vaadin-core-elements are a set of Web Components developed by Vaadin (https://vaadin.com). It focused on developing Web Components for business use cases like data grids, charts, iconsets, file uploaders, and specific user interface elements like a modified dropdown and a date picker.

juicy-ace-editor (https://github.com/Juicy/juicy-ace-editor/)
Custom Element with Ace(http://ace.c9.io/) code editor.

file-saver.js (https://github.com/eligrey/FileSaver.js/)
FileSaver.js implements the HTML5 W3C saveAs() FileSaver interface in browsers that do not natively support it.

saveSvgAsPng (https://github.com/exupero/saveSvgAsPng.git)
Save SVGs as PNGs from the browser.

KaTeX (https://github.com/Khan/KaTeX)
Fast math typesetting for the web.

9.1.2 common-elements

The common-elements package is composed of a set of in-house developed Web Components.

The main focus of this package is to provide panel developers with frequently used functionality. This includes server callbacks, custom input elements, etc.

Currently the Web Components bundled with common-elements are the following:

KaTeX-js Loads the katex.js library, used to render latex math in javascript

auto-update automatically updates server-side data Note that it is very similar to ts-ajax. However ts-ajax only makes a request when you ask for it, auto-update implements an interval with which it will automatically and periodically make a request. Example html in a panel:

```
<auto-update data="{{my_variable}}"
             callback="cpp_callback"
             handle-as="text"></auto-update>
<span>{{my_variable}}</span>
```

color-picker polyfills the html5 color input. This element will be a plain html5 color input element if supported. If not, it will load spectrum.js and provide a color input via JavaScript.

command-input receives three values 'name', 'value', and 'type' and converts it into an appropriate input element. Currently, command-input recognizes the following data types: number, int, long, unsigned int, unsigned long, short, unsigned short, string, double, and float. Examples:

```
<command-input name="dinosaurs are great"
               value="true" type="bool"></command-input>
<command-input name="your_favorite_dinosaur"
               value="deinonychus"
               type="string"></command-input>
<command-input name="positive number"
               value="0"
               type="unsigned int"></command-input>
```

fake-a is a Polymer element that behaves like an anchor (<a>) element, but does not follow the href and thus does not accidentally cause a page refresh. Example:

```
<fake-a on-click="something">click me</fake-a>
```

file-saver-js loads the file-saver.js library. Used to save files with javascript.

iron-flex-layout-attributes provide a simple way to use the css flexbox system. It follows the same syntax as the 'iron-flex-layout', a guide for this syntax is available here: `https://elements.polymer-project.org/guides/flex-layout`

key-value-pair is a simple element that takes a key and a value and presents it nicely. Example:

```
<key-value-pair key="my key"
                value="my value"></key-value-pair>
```

master-detail-layout implements a responsive master-detail layout When on a large enough screen, the master and detail view are displayed side to side. When on a small device, either the master or the detail view is shown, and the user can switch between them Example:

```
<master-detail-layout>
  <div master>I am the master view</div>
  <div detail>I am the detail view</div>
</master-detail-layout>
```

math-equation element that takes latex math input and renders it as a proper equation

moment-js loads the moment.js library

page-js loads the page.js library

relative-time takes a date string and converts it to a relative time (ex: 2h ago) using moment.js Example:

```
<relative-time date="Fri Feb 12 2016 16:30:35 GMT+0100 (CET)">
</relative-time>
```

reset-css makes an element follow the AjaXell theme. When other elements use this element, that element will be enriched with theme directives (colors, sizes, fonts, . . .).

responsive-behavior extends an element with material design breakpoints. It implements the breakpoints from material design. This behavior will give the element a style tag corresponding to the current screen size (extra-small, small, medium, large, extra-large), this can be used to style an element.

save-svg-as-png loads the saveSvgAsPng.js library. It converts an SVG tag to a png bitmap.

ts-ajax makes an ajax request to a specified callback. It is very similar to auto-update, except for the fact that this doesn't have an interval, it only makes a request when you ask it to. Also not that the C++ callback event is 'OnClick' and not 'OnTime' as with auto-update. Example html in a panel:

```
<ts-ajax data="{{my_variable}}"
         callback="cpp_callback"
         handle-as="text"
         auto></ts-ajax>
<span>{{my_variable}}</span>
```

ts-colors Gives an element access to the material design color palette via attributes

ts-tree renders a tree structure from a given JSON.

cytoscape-import loads the cytoscape.js library

candlestick-chart renders a cumulative line chart

cumulative-line-chart renders a cumulative line chart using nvd3.js

cytoscape-import loads the cytoscape.js library

d3-import loads the d3.js library

discrete-bar-chart renders a cumulative line chart

focus-line-chart renders a line chart with focus area using nvd3.js

historical-bar-chart renders a cumulative line chart

horizontal-stacked-bar-chart renders a horizontally stacked bar chart using nvd3.js

line-chart renders a line chart using nvd3.js

multi-chart advanced chart element capable of rendering multiple charts as one

nvd3-chart-behavior the behavior that holds all common element code for NVD3-based chart elements

nvd3-import imports the nvd3.js library, a JavaScript charting library based on D3

parallel-chart renders a cumulative line chart

pie-chart	renders a pie chart using nvd3.js
scatter-chart	renders a scatter chart using nvd3.js
stacked-area-chart	renders a stacked area chart using nvd3.js
stacked-bar-chart	renders a stacked bar chart using nvd3.js
state-diagram	renders a state diagram using cytoscape.js
sunburst-chart	renders a sunburst chart using nvd3.js

This list is taken from `http://cell/ts/common-elements/index.html`

9.1.3 JsonCpp

A panel developer will now use the C++ code primarily for data generation. Therefore it should have some very good tools to send data to the client.

In the legacy codebase the way to put data in JSON format was by using a BOOST Property Tree. Unfortunately BOOST Property trees are not very adequate to generate JSON. They for example don't support the notion of arrays[23].

BOOST Property trees have been replaced in the new codebase by JsonCpp, (`https://github.com/open-source-parsers/jsoncpp`), a lightweight library specifically designed to render and interpret JSON strings.

JsonCpp allow for much more cleaner code. Consider the following example:

This code creates an array of objects using BOOST. Note the fact developers have to render the array manually. This can made code very messy if a developer would need an array inside a property tree, the code stays relatively clean now because the array is the root node.

```
map<string, xdata::Serializable*> dummyParams = getOperation().getParamList();

out << "[";
for ( map<string,xdata::Serializable*>::iterator i = dummyParams.begin();
      i != dummyParams.end(); ++i ) {
  if (i != dummyParams.begin()) {
    out << ",";
  }
  boost::property_tree::ptree object;
  object.put("name", i->first);
  object.put("type", i->second->type());
  object.put("value", dummyParams [i->first]->toString());
  boost::property_tree::write_json(out, deviceTree);
}
out << "]";
```

Now consider the same array of objects, but this time created with JsonCpp. Note that the code is much cleaner.

```
map<string, xdata::Serializable*> dummyParams = getOperation().getParamList();

Json::Value root(Json::arrayValue);
```

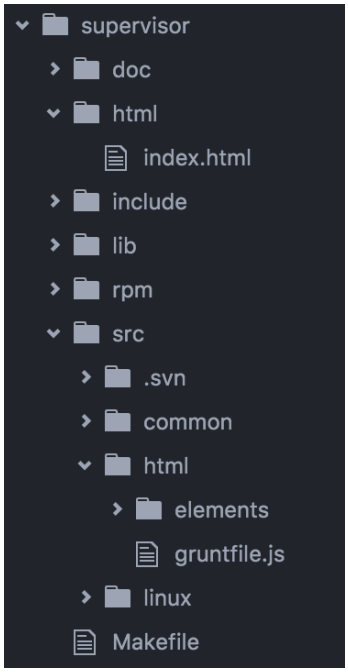


Figure 9.1: Trigger Supervisor folder stucture

```
for ( map<string,xdata::Serializable*>::iterator i = dummyParams.begin();
      i != dummyParams.end(); ++i ) {
  Json::Value input;
  input["name"] = i->first;
  input["type"] = i->second->type();
  input["value"] = dummyParams [i->first]->toString();
  root.append(input);
}
out << root;
```

9.2 New Cell structure

The cell folder structure looks as follows (see figure 9.1). The ‘src/common‘ folder contains all C++ code, except the header files. Header files are kept in the ‘include‘ folder. The ‘src/html‘ folder contains any front-end code that needs processing (e.g. transpiling or minifying code). The result of any processing from this folder will be put in the ‘html‘ folder. The ‘html‘ folder also contains any static front-end code that needs no processing.

For more info about the processing from ‘src/html‘ to ‘html‘, see chapter 9.4.

9.3 Separation of Concerns

SoC is a design primitive, dictating a modular design of the software. This has been implemented in three ways.

Firstly, different syntaxes now are housed in their own files. This allows for significantly less messy code and enables us to implement specific optimizations for each language (for example a CSS pre- and post-processor).

Secondly, the developer is not limited to one source file for each syntax. If circumstances would make some code easier to manage if it is housed across multiple files this is now possible. An example of this would be a panel with multiple specialized sections. Separating these sections will make the code easier to read and maintain.

Thirdly, this approach pushes developers to separate data from markup. This is a very good thing as it causes the code to once again be much more readable. By having the C++ code only produce the necessary data and putting all rendering and interaction on the front-end, developers can also safely replace rendering logic or user interaction flow without having to worry about data generation.

9.4 Grunt build system

During development of the front-end code. Code is kept in multiple files (for the SoC principle).

Instead of loading all the separated files individually at runtime, they will instead be compiled together at compile-time. This will improve loading speeds. The tool used to do this is Grunt `http://gruntjs.com/`, a task runner built on nodeJS that is used to process front-end code languages. It is currently a popular tool to compile, minify, lint, unit-test, etc. front-end code before it is put in production. It has very wide community adoption, which results in a very rich set of tools available for use.

Now that every code language is housed in specialized files, some optimizations on them can run at compile-time. The main objective of these optimizations is to achieve as much browser compatibility as possible.

9.4.1 JavaScript processing

In order to ensure compatibility with all required browsers all JavaScript code is transpiled by Babel `https://babeljs.io/`. This will ensure that newer syntax, like ECMAScript 2016 (ES7), will be transpiled into a more compatible equivalent.

Also the JavaScript code will be transpiled by UglifyJS `https://github.com/mishoo/UglifyJS`. This will implement various code optimizations[24] making the code faster.

9.4.2 CSS processing

SASS

Developers are given the possibility to write SASS code, an extension of the CSS syntax, that will be transpiled into CSS on compile-time using libsass `http://sass-lang.com/libsass`.

Autoprefixer

Also Grunt will automatically add vendor-specific prefixes to CSS properties to maintain the required browser compatibility using a tool called autoprefixer `https://github.com/postcss/autoprefixer`.

9.4.3 HTML processing

No processing is done on the html code except for the fact that it is inlined.

Inliner

Any css link tag or script tag is inlined (i.e. the contents of the referenced file read and inserted in the document) when the url in that tag contains '__inline=true'.

9.5 Templates

To assist developers when creating new interfaces, a set of template files and scripts have been developed.

Where appropriate, a script 'new-element.js' is present. A developer can execute this command to create a new element containing html, sass, and javascript code, all with inline documentation and a demo page.

9.6 Panel registration system

An interface consists of multiple libraries. Each of these could possibly contain element definitions (e.g. the AjaXell library has elements concerning things like session management).

To allow any loaded library to declare its own Web Components, a panel registration system is included in the AjaXell libary.

It allows a developer to register any custom elements he developed in his cell like so:

```
1 void <package-name>::Cell::init()
2 {
3   getContext()->addImport("/<package-path>/html/elements/elements.html");
4 }
```

With the 'elements.html' file containing element definitions, either in eager loading or in lazy loading syntax.

9.6.1 Eager loading and lazy loading

Depending on the content of the 'elements.html' file, the element definitions will either be loaded eagerly (i.e. at page load time) or lazy (i.e. when they are needed). Of course a developer is encouraged to implement the lazy loading approach, as it will decrease initial page loading times.

The eager loading approach is a list of HTML imports.

```
1 <link rel="import" href="my-first-element/my-first-element.html" >
2 <link rel="import" href="my-second-element/my-second-element.html" >
```

While the lazy loading approach consists of a custom function provided by AjaXell.

```
<script>
var base = document._currentScript.baseURI;
base = base.substring(0, base.lastIndexOf("/") + 1);

LazyLoad({
  base: base,
  provides: {
    "my-first-element": "my-first-element/my-first-element.html",
    "my-second-element": "my-second-element/my-second-element.html"
  }
})
</script>
```

Chapter 10

Documentation

Documentation is something commonly taken too lightly. The legacy panel system contains little to no documentation. This frustrates developers and inhibits any changes to the codebase, because there might be this undocumented flow of code that will break and only found out about much later.

Fortunately there are some tools not only to properly write documentation , but also to encourage developers to write documentation as the codebase evolves.

10.1 Inline documentation

Documentation regarding the description code itself is kept close to the code, in the form of inline documentation.

This means most of the documentation will be housed along with with the source code itself. The goal is to minimize separation of code and documentation as this easily leads to inconsistencies between the two.

Advantages of inline documentation are the reduced chances for outdated documentation and being able to enrich source code with typed annotations [25].

Source code consists of C++, JavaScript, HTML, and CSS code. The inline documentation described here is applicable to the last three.

10.1.1 JSDocs

The syntax used to document JavaScript code is called JSDocs and is currently at version 3[25][26]. It provides us with a rich set of expressions enabling a developer to write documentation comparable to JavaDoc.

JSDocs has wide industry adoption for JavaScript projects. It is widely used to make code more understandable, generate HTML documentation, or use it to generate the large traditional developer manuals.

In addition to JSDocs there are specific points in the source code where a developer can provide code examples and extra directives to document HTML and CSS code. This is however a non-standard method since there is no standardized way to document any of the other languages inline.

10.2 Global level

The global level is the highest level and is the only level where documentation is separated from the source code.

10.2.1 Goals

The main purpose of this documentation level is to be an entry point for developers. It will teach developers the basics of the codebase, why things need to be done one way or the other, and will point them to lower-level documentation whenever appropriate (such as which package or element could be useful for a particular use case).

Where lower level documentation will only focus on how to get stuff done, this documentation level also has the responsibility to show developers concepts like Separation of Concerns (see chapter 9.3) and modular thinking.

10.2.2 Sphinx

This global documentation level is built using Sphinx (`http://www.sphinx-doc.org/`). It takes a set of wiki-like documents and converts them in various types of resources (HTML, LaTeX, ...). This is primarily focused on HTML output, but the LaTeX version is included in this document as appendix A.

10.3 Package level

The global level gives an overview of the packages that are available to a panel developer. The package level lies under the global level and is the first automatically generated level. It describes the package's content and its capabilities.

10.3.1 Goals

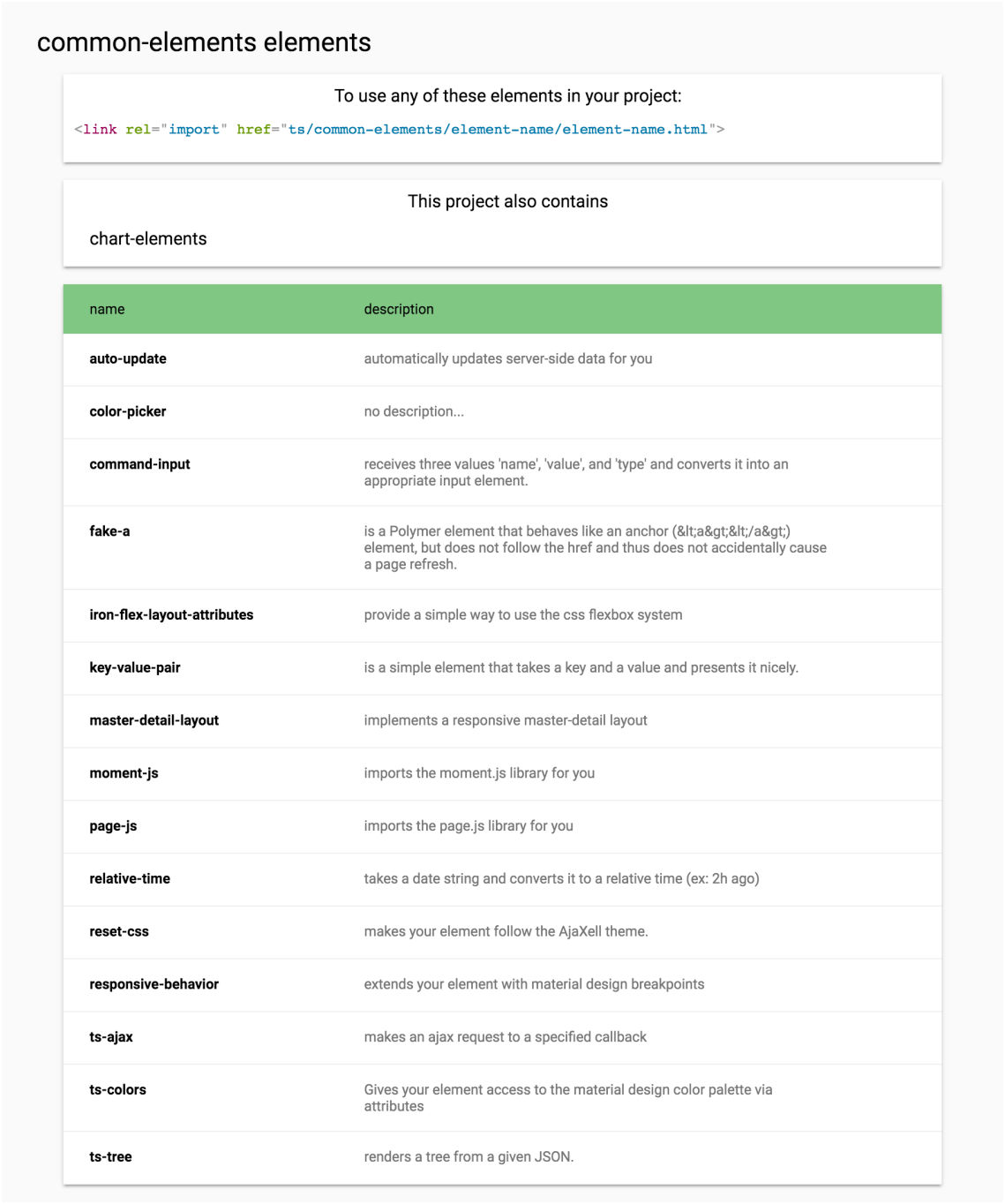
Primarily, this documentation level gives a quick overview of the content of a package. It also points readers to additional resources like element-level documentation, the repository where the source code is hosted, and live demos where supported.

10.3.2 Grunt

This documentation is generated in the Grunt build cycle described in chapter 9.4. It loads and interprets every component of the package and generates a summary page giving a general overview and pointing to several useful resources for each component such as the documentation on the element level, a link to the code repository, and a link to a live demo of the component if available.

The code it uses to render this documentation is housed in the source code of each component. It gets interpreted by Grunt and is then compiled in the package documentation page.

An example of a package level documentation page is show in figure 10.1.



common-elements elements

To use any of these elements in your project:

```
<link rel="import" href="ts/common-elements/element-name/element-name.html">
```

This project also contains

chart-elements

name	description
auto-update	automatically updates server-side data for you
color-picker	no description...
command-input	receives three values 'name', 'value', and 'type' and converts it into an appropriate input element.
fake-a	is a Polymer element that behaves like an anchor (<a>) element, but does not follow the href and thus does not accidentally cause a page refresh.
iron-flex-layout-attributes	provide a simple way to use the css flexbox system
key-value-pair	is a simple element that takes a key and a value and presents it nicely.
master-detail-layout	implements a responsive master-detail layout
moment-js	imports the moment.js library for you
page-js	imports the page.js library for you
relative-time	takes a date string and converts it to a relative time (ex: 2h ago)
reset-css	makes your element follow the AjaXell theme.
responsive-behavior	extends your element with material design breakpoints
ts-ajax	makes an ajax request to a specified callback
ts-colors	Gives your element access to the material design color palette via attributes
ts-tree	renders a tree from a given JSON.

Figure 10.1: Documentation page of the common-elements package

10.4 Element level

The lowest level of documentation is documentation of individual web components. This level is also auto-documented from the component's source code. But unlike the documentation on the package level, where documentation is generated on compile time, the documentation here is rendered on the fly, client-side.

10.4.1 Goals

This documentation provides an overview of all the properties and available calls of this component. It can also provide code examples and even live demos.

10.4.2 iron-component-page

Client-side rendering is done by using a specialized web component called 'iron-component-page' (`https://elements.polymer-project.org/elements/iron-component-page`). It uses the 'hydrolysis.js' library to interpret the inline documentation provided by the developer in the source code of the web component, and compiles this into a documentation page.

An example of an element level documentation page is show in figure 10.2.

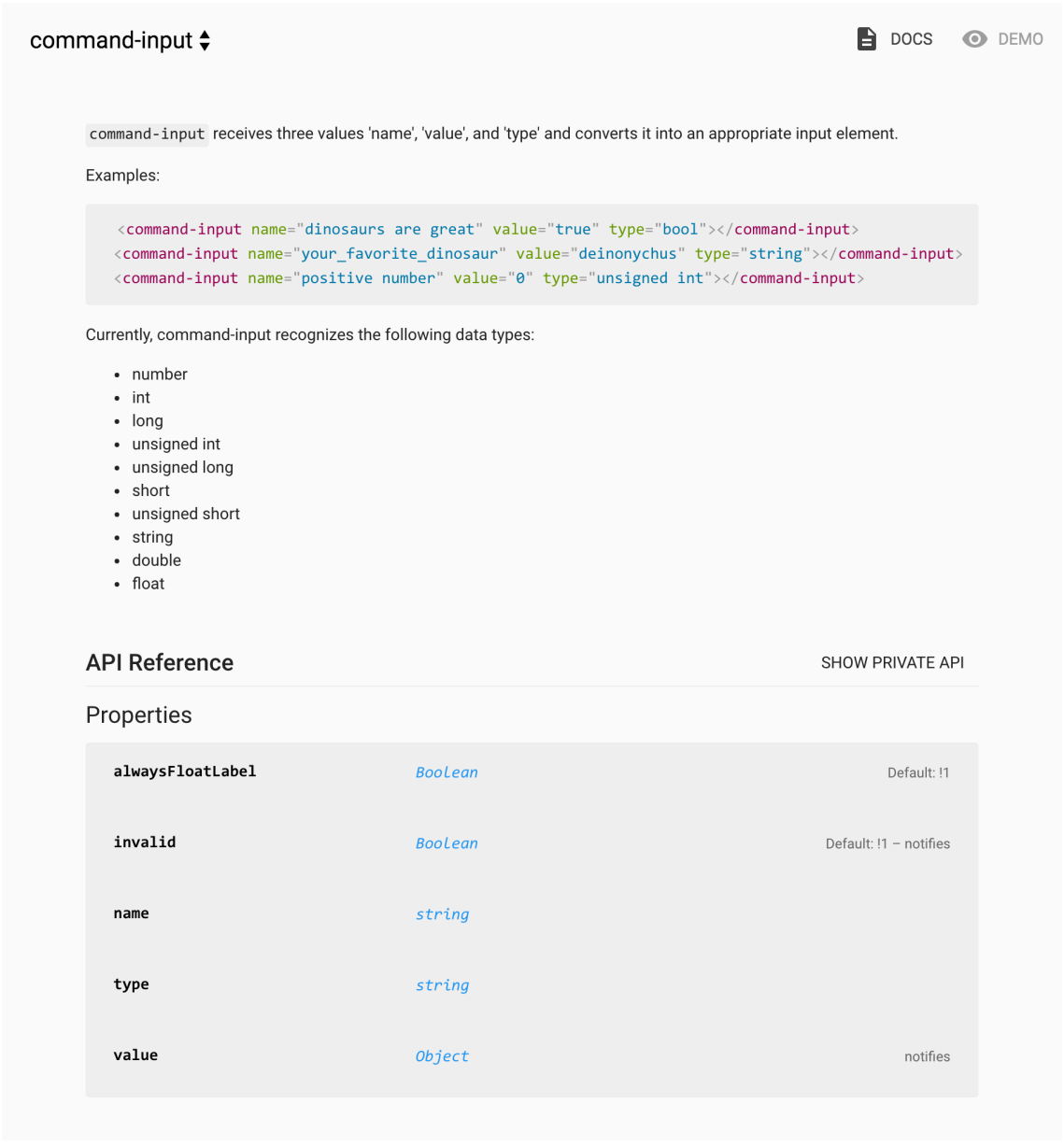


Figure 10.2: Documentation page of the command-input element

This page is intentionally left almost blank

Chapter 11

Browser testing

The TS interface officially supports the latest ESR release of Mozilla Firefox. However, user tend to use many different browsers. When developing new interfaces it is very unpractical to test all these browsers manually and consistently.

Also, because of the frequent changes made to web browsers (described in chapter 3.1.1), it has become very important to test interface functionality as browser versions get updated at production systems.

11.1 Selenium

Selenium is a tool that can automate a browser. Its most common uses are to perform tests or to automate web interfaces.

Starting from version 2, it uses an open standard called ‘WebDriver‘ to interact with a browser. Most modern browsers have a native implementation of this standard, and separate drivers exist for browsers that do not (including IE6).

Selenium can run under Windows, Mac OS X, and Linux (Debian & RHEL). It has official libraries for C, Haskell, Java, JavaScript (Node.js), Objective-C, Perl, PHP, Python, R, and Ruby.

11.2 Web-component-tester

The polymer project contains a dedicated testing tool to test Polymer elements and is used in the TS.

It allows a developer to make a series of tests for every Polymer element (and thus every interface). These tests are performed using the grunt build system when executing ‘grunt test‘.

Tests are defined in a ‘test‘ folder in the source folder of every element. There a developer can define tests with the ‘test()‘ function like this:

```
1 // this function tests if the Polymer element has declared an object
2 // 'someObject' and it has a property 'name' with value 'deinonychus'
3 test('defines the "author" property', function() {
4     assert.equal(element.someObject.name, 'deinonychus');
5 });
```

```
Running "wct-test:local" (wct-test) task
Installing and starting Selenium server for local browsers
Selenium server running on port 65482
Web server running on port 2000 and serving from /Users/glenn/l1ce
chrome 53               Beginning tests via http://localhost:2000/components/l1ce/generated-index.html?cli_browser_id=0
chrome 50               Beginning tests via http://localhost:2000/components/l1ce/generated-index.html?cli_browser_id=1
safari 9.1.1            Beginning tests via http://localhost:2000/components/l1ce/generated-index.html?cli_browser_id=3
firefox 46              Beginning tests via http://localhost:2000/components/l1ce/generated-index.html?cli_browser_id=2
chrome 53               Tests passed
safari 9.1.1            Tests passed
chrome 50               Tests passed
firefox 46              Tests passed
Test run ended with great success

chrome 53 (6/0/0)                   chrome 50 (6/0/0)
firefox 46 (6/0/0)                  safari 9.1.1 (6/0/0)

Done.
```

Figure 11.1: Console output when running tests with web-component-tester

```
6  // tests if the function 'sayHello()' returns a specific string.
7  // also tests if the function 'sayHello()' respects its arguments.
8  test('says hello', function() {
9      assert.equal(element.sayHello(), 'template-element says, Hello World!');
10     var greetings = element.sayHello('greetings Earthlings');
11     assert.equal(greetings, 'template-element says, greetings Earthlings');
12 });
```

More advanced use cases, such as testing AJAX (Asynchronous JavaScript And XML) requests or even emulating an AJAX response are also possible. More detailed examples of web-component-tester can be found in the Sphinx documentation, which can be found in appendix A.

When a developer executes ‘grunt test‘, a Selenium server is started and the defined tests are performed using the latest version of Mozilla Firefox, Google Chrome, Google Chrome Canary, and Safari (if possible).

Chapter 12

Results

12.1 Loading times

Table 12.1 shows an overview of the initial full page loading times for the legacy TS (version 2.1.0) and the new TS (version 3.4.0). That is, a page load from a new browser tab with all caches removed.

This test is performed with the timeline panel of Google Chrome 50.0.2661.86 (64-bit).

It is expected that the TS 3.x has higher values for everything in this table, because it loads two front-end libraries (Dojo & Polymer).

Notable is the decrease of scripting time for the TS 3.x relative to the TS 2.x. This is because Dojo is minified and packaged into one JavaScript file in the TS 3.x release, where as in the TS 2.x release it was not. Also, because this test is performed in Google Chrome, which has native support for Web Components, very little scripting needs to be done. This result will be different in other browsers like Mozilla Firefox, where Web Components support needs to be emulated. Then again, the lazy loading system largely removes this overhead from the initial page loading time, so only minor differences would be expected here.

Rendering time has increased the most going from TS 2.x to TS 3.x. This makes sense as Polymer renders everything on the front-end, whereas Dojo used to render everything server-side. During initial page load this rendering load is primarily caused by the rendering of the left side menu. The increase of painting time follows the same logic as the rendering time.

Also notable is the increase of idle time. This means that the browser needs to wait for a task to

	TS 2.1.0 (Dojo)	TS 3.4.0 (Dojo + Polymer)	difference	difference (%)
Loading	44.5ms	112.4ms	+67.9ms	+152.58%
Scripting	1227.6ms	1187.6ms	-40ms	-3,26%
Rendering	29.7ms	171.0ms	+141.3ms	+475,76%
Painting	7.5ms	36.1ms	+28.6ms	+381.33%
Other	106.4ms	335.9ms	+229.5ms	+215,69%
Idle	213.6ms	775.1ms	+561.5ms	+262,87%
Total	**1.63s**	**2.62s**	**+990ms**	**+60,73%**

Table 12.1: Page loading times for TS 2.x and 3.x

finish before it can start another. This is caused because the TS 3.x loads the default panel after the initial page load. Which means the TS makes extra network request, to fetch an interface panel, right after initializing. This is counted with the initial page load. TS 2.x just shows a blank page, it loads no default panel. Because the browser needs to wait for the extra network requests to finish before it can render the default panel, the idle time goes up by a lot.

In total, the initial page loading time increased with about 60%, which is an acceptable increase given the new TS runs 2 libraries concurrently.

12.2 CPU consumption

Both TS releases have negligible CPU usage when doing a fresh page load, and stay at 0% CPU usage when the user is not interacting with the system.

TS 3.x uses hardware acceleration for it's animations since they are all made using CSS transform properties or using Web Animations[27]. The only exception to this is the 'paper-spinner' element. Which displays a loading animation. The TS 2.x release did not have any animations.

12.3 Memory consumption

Chapter 6.7.2 described the memory leak problems in TS 2.x. It showed a clear memory leak problem that needs addressing in TS 3.x.

Image 6.6 showed that the new interface contains no memory leaks, unless of course a panel developer creates one. This is why the 'ts-ajax' and 'auto-update' elements in the 'common-elements' package have been equipped with ways to detect a circular reference, as they are the most likely to be used in one.

Unfortunately it also showed in figure 6.5 that legacy panels in the new TS still suffer from this memory leak. This is because the circular references causing the memory leak reside in the Dojo library itself, and thus would be impractical to address. Therefore, any interface that included auto-refreshes had the highest priority to be converted to a new TS 3.x interface.

Because TS 3.x uses client-side interface rendering rather than server-side as the TS 2.x did, it uses more memory from the browser.

Chapter 12.1 already described that in TS 3.x the memory used by an interface panel will be released after it switches to another panel. It also described that in TS 2.x the memory consumption grows linearly with the amount of panels loaded by the user.

To test the difference in memory consumption, both TS versions were opened in a new tab while memory consumption is monitored. No panels are loaded, the interfaces are just left for 120s. The mean memory consumption in those 120s is then taken as the mean memory consumption for that TS release. The results of this test are shown in table 12.2.

12.4 Functionality

TS 3.x has functionally more capabilities for the interface than TS 2.x had. More importantly, the TS interface is now no longer bound to one framework. Any Web Component can be used, and extra

	Google Chrome	Firefox
TS 2.1.0 (Dojo)	20.051MB	7.06MB
TS 3.4.0 (Dojo + Polymer)	24.564MB	10.96MB
difference	+4.513MB	+3.9MB
difference (%)	**+22.51%**	**+55.24%**

Table 12.2: Memory usage for TS 2.x and 3.x in Mozilla Firefox and Google Chrome

functionality can be developed in-house. This unlike TS 2.x where developers were functionally bound to the elements the Dojo developers provided.

This makes TS 3.x far more easy to change, and thus more ready for the future.

12.5 SDK improvements

The fact that multiple programming languages are no longer placed into one file, but distributed across multiple files, makes the developing an interface panel a lot easier.

The Web Components approach to build interfaces gives developers a set of powerful tools that are easy to use and extend.

12.6 Developed panels

The Control Panels are a set of custom interfaces, developed for an individual cell. The other panels however occur on every cell. And are upgraded as part of the new TS release.

12.6.1 Commands

The new commands panel use the ‘command-input‘ element for its input. Making it easily extendible to understand more input types (e.g. vectors). Currently it understands number, int, long, unsigned int, unsigned long, short, unsigned short, string, double, and float input.

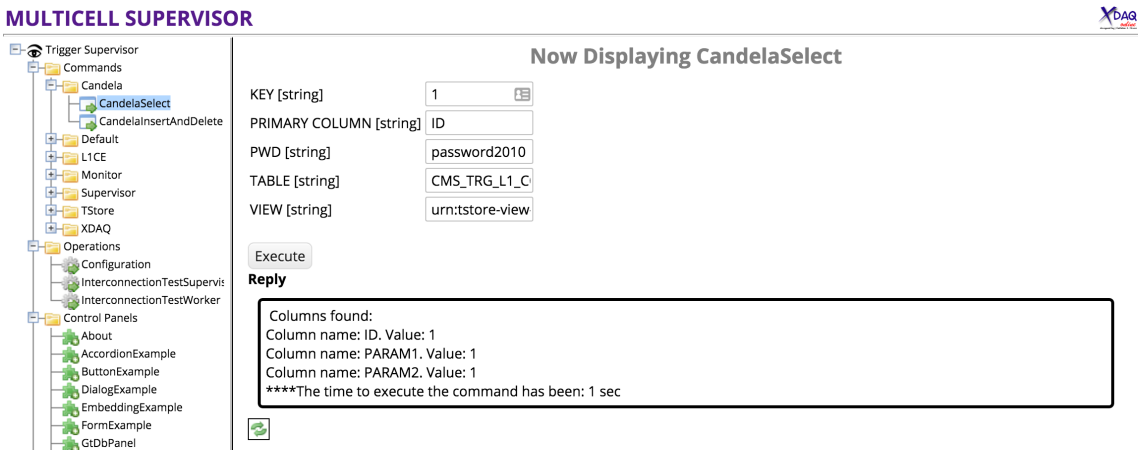


Figure 12.1: TS 2.x commands panel

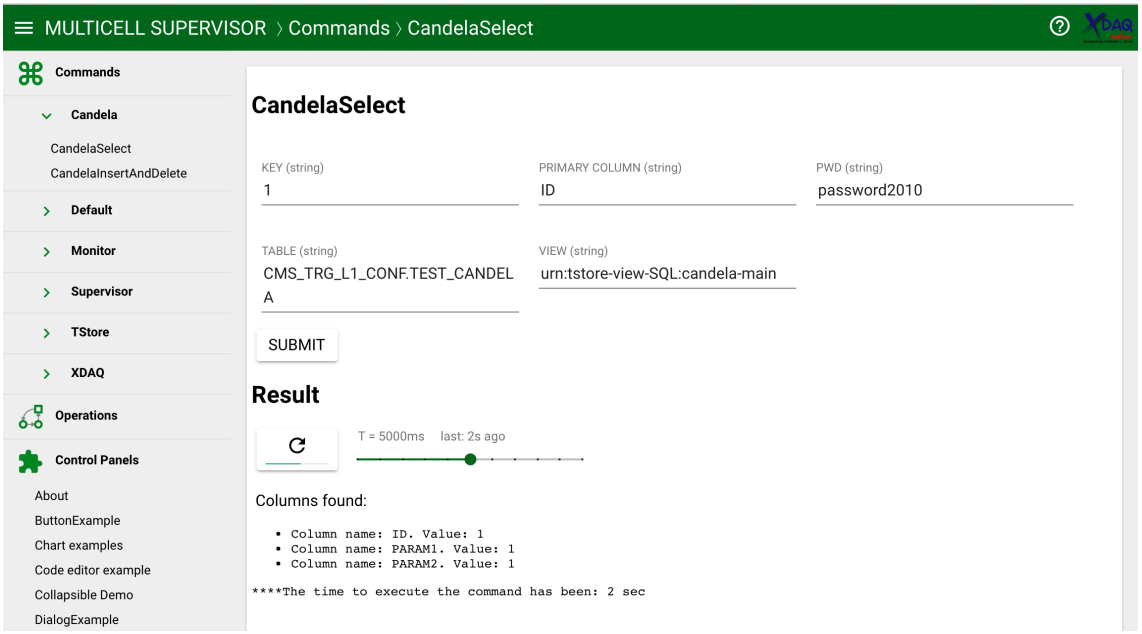


Figure 12.2: TS 3.x commands panel

12.6.2 Operations

The TS 2.x operations panel had some problems with auto-updating. The state diagram tended to update very late, if it updated at all. Result data and new available commands usually took more than 10 seconds to show up in the interface.

The new operations panel is now far more responsive. The state diagram is available when clicking on an icon, as it was deemed a waste of space to show it by default.

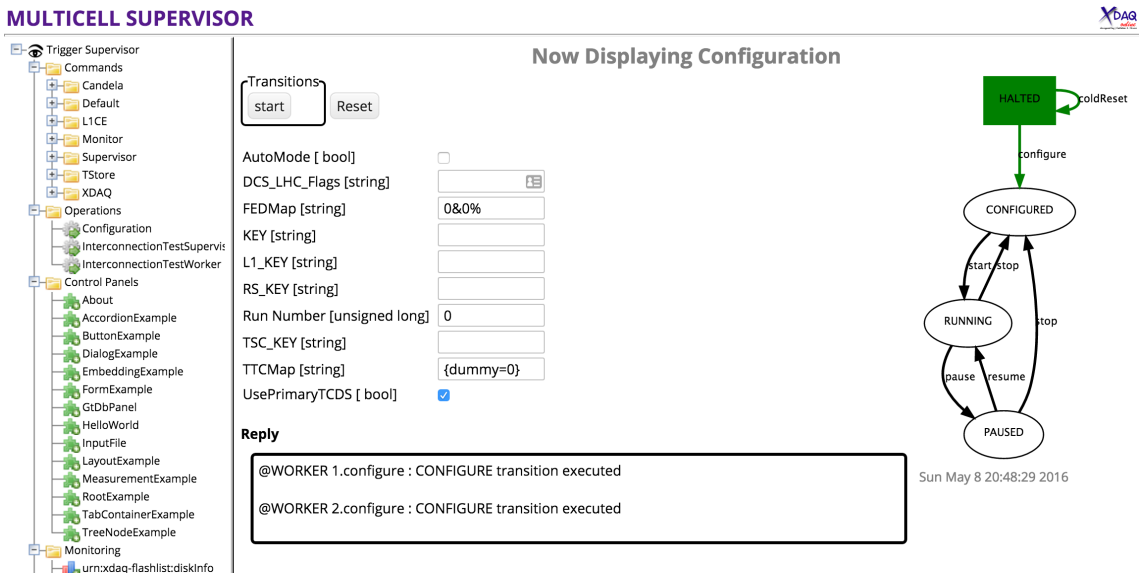


Figure 12.3: TS 2.x operations panel

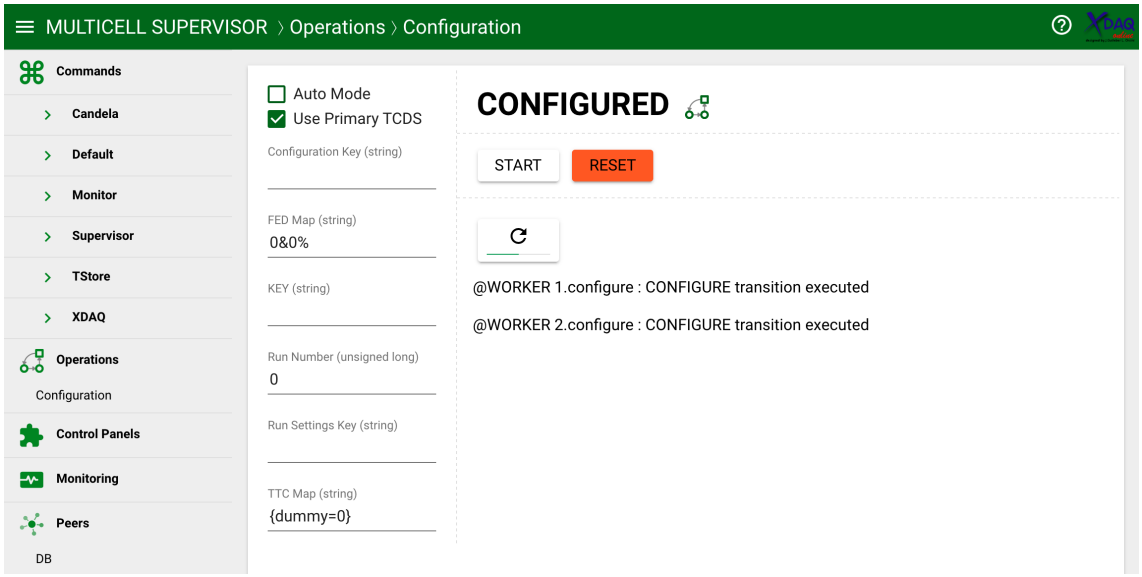


Figure 12.4: TS 3.x operations panel

12.6.3 Flashlists

A flashlist is a more abstract interface designed to display tabular data. This data can change with a regular interval. A table cell can contain a string, date, number, or another table.

The flashlist panels now have a user-configurable auto-update function. The flashlist can deploy custom renderers in the table depending on the data type, for example a date will be shown as relative time (e.g. 9 minutes ago), instead of just showing a time stamp. This list of custom renderers can be extended easily.

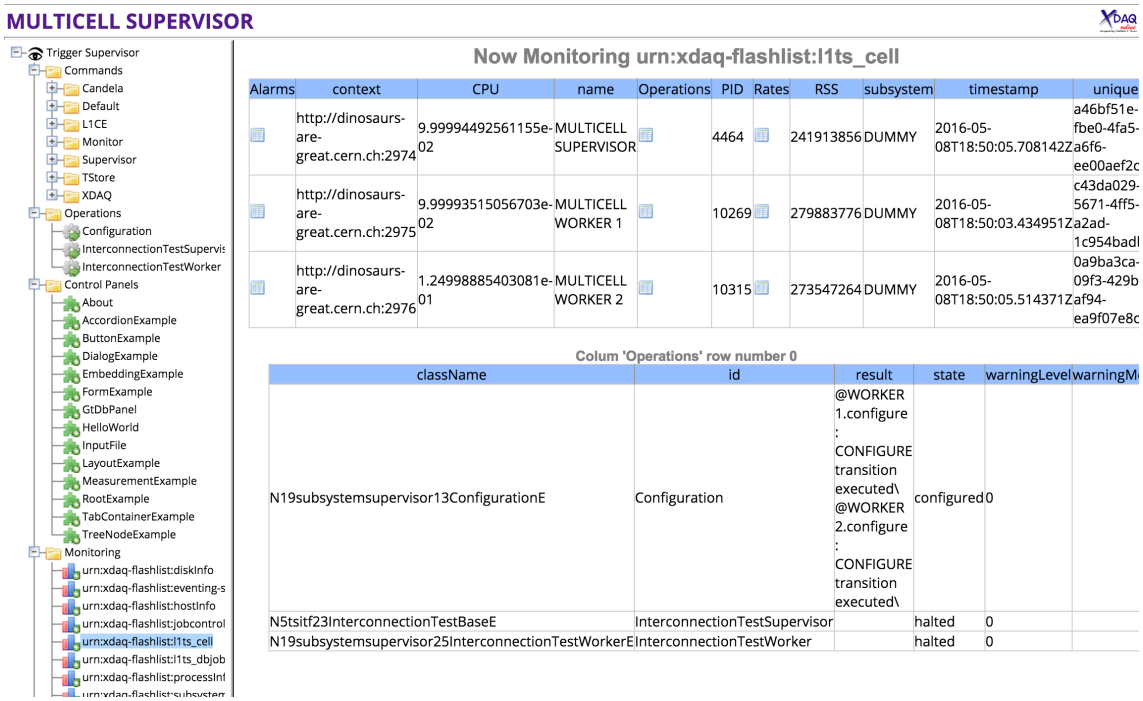


Figure 12.5: TS 2.x flashlists panel

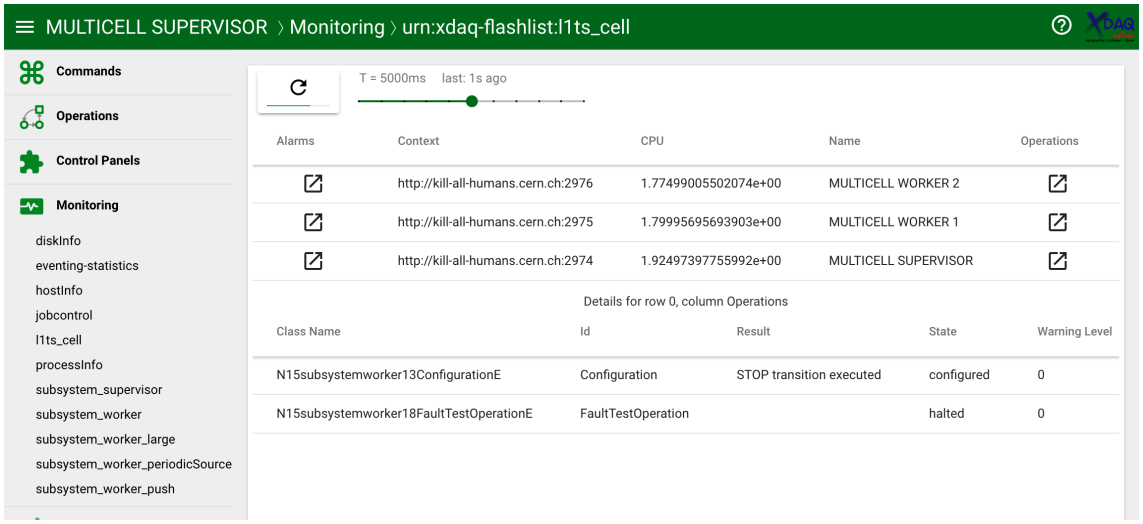


Figure 12.6: TS 3.x flashlists panel

Chapter 13

Future

A lot of big software projects (such as the Level-1 Configuration Editor and the Level-1 page) rely on the TS. Some of these projects are starting to benefit from the changes made to the TS, and some of them are scheduled for a complete overhaul using the TS redesign as a template.

Some things did not make it into the TS, because of backward-compatibility problems or time issues. However, as time goes on, the need to keep compatibility with legacy systems will fade away. And some improvements may yet become possible.

13.1 Dojo-free TS release

At some point, all legacy (Dojo) panels will have been migrated to Polymer. When this has occurred, legacy code can be removed from the TS.

A lot of code can be removed. The Dojo component classes, legacy session logic, the legacy event system, etc.

Furthermore, Dojo can be removed from the front-end interface. This is expected to bring a noticeable speed improvement to the initial page load.

At this point, the TS can also be prepared to take on a next framework, where this time Polymer will be considered legacy code.

13.2 HTML5 WebSocket

A WebSocket is a full-duplex HTTP-like connection between a web browser and a web server. Both the client and the server must support this protocol before such a connection can be set up.

This allows for very efficient communication between client and server, and will be especially useful when the server has frequently changing data to serve to the client. Traditionally this has been achieved using various sort of polling, which puts unnecessary load on the server and the network.

This connection will also allow the web browser to receive updates near-instantaneous.

Currently, the most CPU intensive tasks in the TS interface belong to the auto-update logic. This would be drastically reduced when implementing WebSockets.

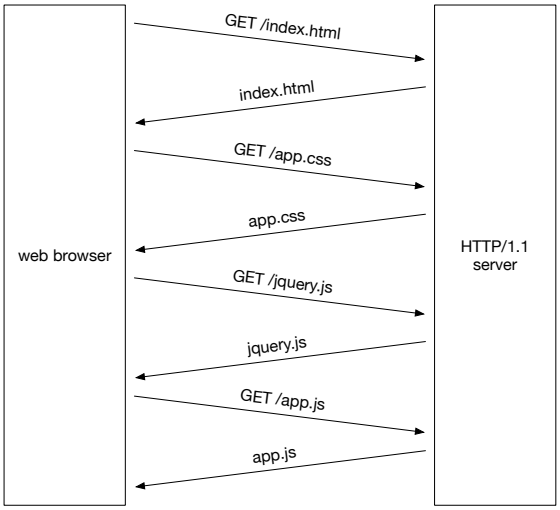


Figure 13.1: A common request/response diagram when using a HTTP/1.1 server

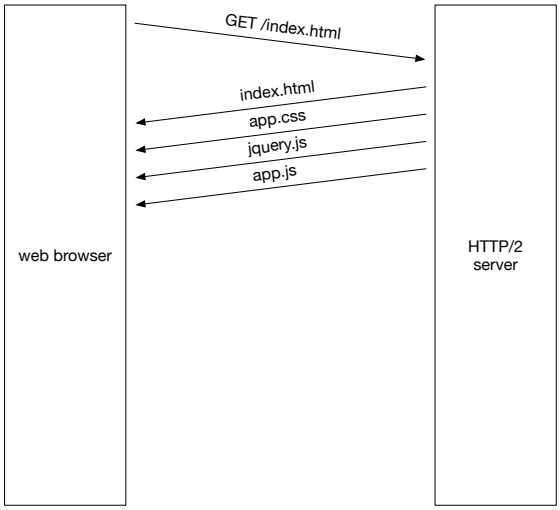


Figure 13.2: A common request/response diagram when using a HTTP/2 server with Server Push

13.3 PRPL

The PRPL[28] pattern is a software pattern for web apps designed by Google and stands for:

- **Push** critical components for initial page load
- **Render** the initial page
- **Pre-cache** other components of the interface
- **Lazy-load** needed components

It uses Web Components, HTML Imports, Service Workers, and HTTP/2 to accomplish all these. The TS already uses two of them, Web Components and HTML Imports. The two others that didn't make it into the TS are explained here in a bit more detail.

13.3.1 HTTP/2 Server Push

Using HTTP/2, the server can interpret requested resources and decide to not only return the requested resource to the client, but also provide the user with additional resources related to the requested resource.

This is useful on the first page load, where a web browser requests the initial page (usually 'index.html'). This file very often contains references to other resources such as CSS or JavaScript files. Normally the web browser needs to make another request for each of these resources. HTTP/2 can multiplex these related resources along with the originally requested resource over the same connection. This severely reduces network latencies, as everything is returned in one payload. This effect is demonstrated in image 13.1 and 13.2.

13.3.2 HTML5 Service Worker

A Service Worker is a JavaScript file that is run in the browser as a separate thread. Unlike traditional JavaScript files, this file has no access to the DOM or any global variables like 'document'

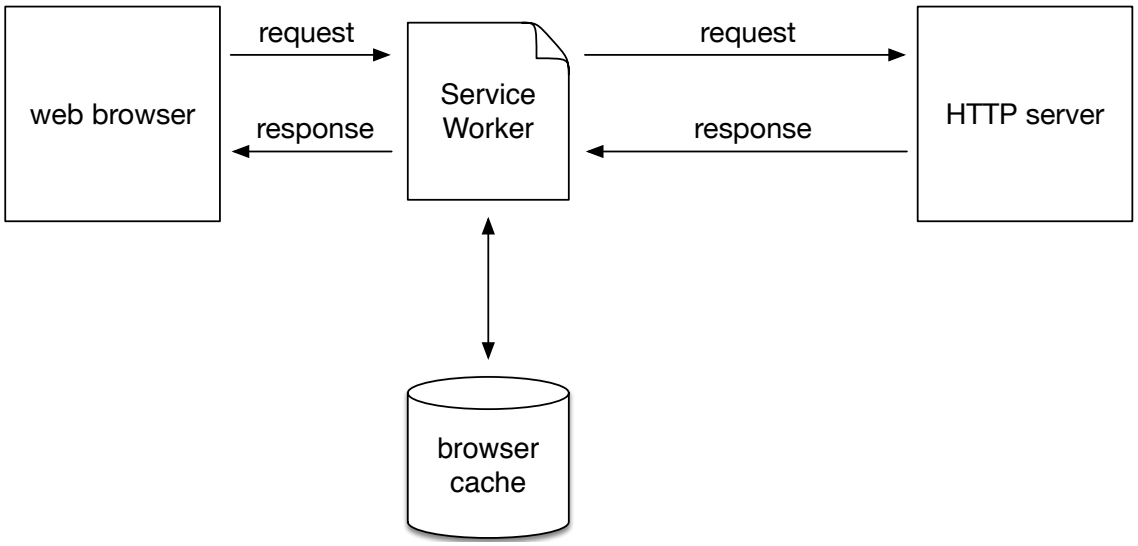


Figure 13.3: Logical location of the Service Worker in a web browser

or ‘window‘. This file runs on a different scope.

A Service Worker runs in between the browser and the network. It is able to intercept requests and modify them. It can even provide a response, thereby completely bypassing the server and the network.

It also has full control over the browser cache. So the interface can programmatically control what is put in the cache and when it is served or renewed. This is shown visually in image 13.3.

This enables the interface to do two thing that were previously impossible, but very useful.

The first is pre-caching. When the initial page load is done, the Service Worker can silently pull resources from the web server before they are actually needed. This will make interacting with the application a lot faster.

The second thing is offline functionality, and is a more elaborate version of pre-caching. When enough resources are pulled into the cache, the interface does not need the web server anymore: in some cases (including the initial page load), the interface can work without an internet connection.

This will completely remove the need for network requests (except for receiving new data): static resources (and data) are kept client-side and do not put a load on the network anymore, vastly increasing performance.

This page is intentionally left almost blank

Chapter 14

Conclusion

The main objective was to upgrade the TS to be able to provide more advanced interfaces, and to keep compatibility with legacy interfaces.

The new interface engine has achieved 100% backwards compatibility, while providing a completely new way to develop new interfaces.

This new interface engine and can be easily extended and is ready for any future use-cases as it is built to change. The developers are not bound to the functionality of one framework, rather it is build on open standards and thus ensures maximum compatibility with future technologies.

The interface developers now have internal, semi auto-generated, documentation at their disposal and have an active community on the world wide web to fall back to.

This page is intentionally left almost blank

Bibliography

[1] T. McCauley and L. Taylor, "CMS Higgs Search in 2011 and 2012 data: candidate ZZ event (8 TeV) with two electrons and two muons," Jul 2012, cMS Collection. [Online]. Available: https://cds.cern.ch/record/1459462

[2] T. McCauley and L. Taylor, "CMS Higgs Search in 2011 and 2012 data: candidate photon-photon event (8 TeV)," May 2013, cMS Collection. [Online]. Available: https://cds.cern.ch/record/1459459/

[3] D. Barney, "CMS slice image with transverse/longitudinal/3-D views," Nov 2011, cMS Collection. [Online]. Available: https://cms-docdb.cern.ch/cgi-bin/PublicDocDB/ShowDocument?docid=5697

[4] A. Breskin and R. Voss, *The CERN Large Hadron Collider: Accelerator and Experiments*. Geneva: CERN, 2009.

[5] L. Taylor, "Detector overview | cms experiment," http://cms.web.cern.ch/news/detector-overview, 2011.

[6] T. C. Collaboration, "The cms experiment at the cern lhc," *Journal of Instrumentation*, vol. 3, no. 08, p. S08004, 2008. [Online]. Available: http://stacks.iop.org/1748-0221/3/i=08/a=S08004

[7] D. Abbaneo, A. Bal, P. Bloch, O. Buchmueller, F. Cavallari, A. Dabrowski, P. de Barbaro, I. Fisk, M. Girone, F. Hartmann, J. Hauser, C. Jessop, D. Lange, F. Meijers, C. Seez, W. Smith, *The Compact Muon Solenoid Phase II Upgrade Technical Proposal*. European Organization for Nuclear Research (CERN), 2015.

[8] J. Brooke, K. Bunkowski, I. Cali, C. G. Larrea, C. Lazaridis, and A. Thea, "SWATCH: common control SW for the uTCA-based upgraded CMS L1 Trigger," *J. Phys.: Conf. Ser.*, vol. 664, no. 8, p. 082012. 8 p, 2015. [Online]. Available: https://cds.cern.ch/record/2134631

[9] E. S. Raymond, *The Cathedral and the Bazaar*. O'Reilly & Associates, Inc., 1999, ch. 3, pp. 38–44.

[10] kangax, webbedspace, zloirock, "Ecmascript 6 compatibility table," http://kangax.github.io/compat-table/es6/, 2016.

[11] Tab Atkins Jr., Elika J Etemad, "Css scoping module level 1," https://drafts.csswg.org/css-scoping/, 2016.

[12] I. M. de Abril and C.-E. Wulz, "The CMS Trigger Supervisor: Control and Hardware Monitoring System of the CMS Level-1 Trigger at CERN," Ph.D. dissertation, Barcelona, Autonoma U., 2008. [Online]. Available: http://cds.cern.ch/record/1446282

[13] S. G. Raymond Eric S, *The Jargon File, Version 4.2.2, 20 Aug 2000*, 1 2002.

[14] S. R. Marcin Kalicinski, “How to populate a property tree,” https://larseidnes.com/2014/11/05/angularjs-the-bad-parts/, 2008.

[15] World Wide Web Consortium (W3C), “Custom elements,” https://w3c.github.io/webcomponents/spec/custom/, 2015.

[16] Mozilla Developer Network, “Web components,” https://developer.mozilla.org/en-US/docs/Web/Web_Components, 2014.

[17] C. House, “Html5 web components: The solution to div soup?” https://www.pluralsight.com/blog/software-development/html5-web-components-overview, 2015.

[18] J. L. C. R. J. W. Matias Duarte, Nicholas Jitkoff, “Material design principles,” https://www.google.com/events/io/io14videos/79edef8b-96d4-e311-b297-00155d5066d7, 2014.

[19] Google, “Material design,” https://www.google.com/design/spec/material-design/, 2014.

[20] IBM DeveloperWorks, “Memory leak patterns in javascript,” http://www.ibm.com/developerworks/library/wa-memleak/, 2007.

[21] J. Sutherland and J. Sutherland, *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Business, 9 2014. [Online]. Available: http://amazon.com/o/ASIN/038534645X/

[22] P. Authors, “Polymer 1.0,” https://www.polymer-project.org/1.0/docs/, 2016.

[23] L. Eidnes, “Angularjs: The bad parts,” http://www.boost.org/doc/libs/1_55_0/doc/html/boost_propertytree/parsers.html, 2014.

[24] M. Bazon, “Uglifyjs — the compressor,” http://lisperator.net/uglifyjs/compress, 2012.

[25] Google Developers, “Annotating javascript for the closure compiler,” https://developers.google.com/closure/compiler/docs/js-for-compiler, 2016.

[26] “@use jsdoc,” http://usejsdoc.org/, 2011.

[27] A. D. T. A. Brian Birtles, Shane Stephens, “Web animations,” https://w3c.github.io/web-animations/, 2016.

[28] P. Authors, “Serve your app,” https://docs2-dot-polymer-project.appspot.com/1.0/toolbox/server, 2016.

Appendix A

Sphinx Documentation

Most part of the documentation of this project is auto-generated. The manual documentation, developed in Sphinx (described in chapter 10.2.2), is enclosed as an appendix to this document.

This page is intentionally left almost blank

CMS Trigger Supervisor Software Documentation

Release 3.4.0

Glenn Dirkx

May 24, 2016

CHAPTER

ONE

BUILDING YOUR OWN CELL PANELS

1.1 Welcome

This documentation will show you how to develop panels.

1.1.1 Scope of this document

This document contains both basic info and quickstarters, but also very detailed descriptions of the inner workings of the technologies used.

All examples are taken from the Subsystem Supervisor unless otherwise specified. (https://svnweb.cern.ch/trac/cactus/browser/trunk/cactusprojects/subsystem/supervisor)

1.2 Quickstart section

1.2.1 Setting up the Cell

Starting from this point we will assume you already have a working cell and you now have arrived at the point you wish to develop panels for it.

Front-end code (HTML, CSS, and JavaScript) have a separate build cycle, separate from the makefile. These are the steps necessary to setup this build system.

Making your life easier

We have copied the files that will be created in this page into a tarball. This will absolve you from having to create any files.

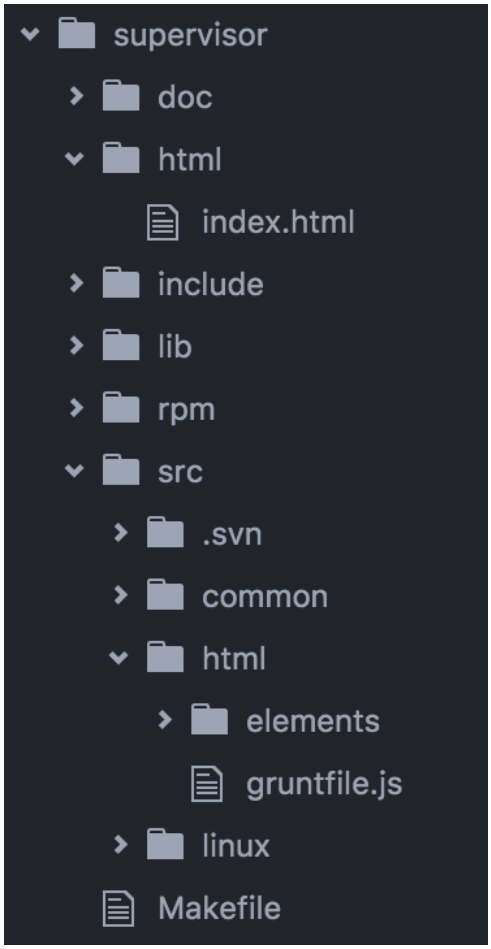
To use it, run:

```
svn export svn+ssh://svn.cern.ch/reps/cactus/trunk/cactuscore/ts/doc/cell-skeleton.tar
tar -xzvf cell-skeleton.tar
```

Now you can continue following this tutorial, but you don't have to create files anymore.

Making directories

Extend the file structure of your cell to include the following folders and files:



The html folder will contain the build output of the source files in the /src/html folder.

If you have static resources you wish to serve in your panel, put them in the /html folder. /src/html is only meant for source files that need to be processed in some way.

Setting up Grunt

A panel is composed of different code languages, namely HTML, CSS, and JavaScript. When you develop a panel, each of these languages are housed in their own files.

This makes things easier for you to read and allows your editor to use code highlighting and syntax checkers.

It also allows us to perform optimizations on your code. The JavaScript will be optimized and the SASS code will be compiled into CSS and enhanced for compatibility.

The build system will put all generated files into the /html folder.

Grunt (http://gruntjs.com/) is the tool we'll use to accomplish all this.

if you do not have npm installed, run

```
sudo yum install -y npm
```

now run

```
cd src/html
sudo npm install -g grunt-cli
```

This will install the grunt command line tools on your system, you only need to do this once since it's a global install (the -g option).

Put this in the /src/html/gruntfile.js file

/src/html/gruntfile.js

```
require('es6-shim');
module.exports = function(grunt) {
    grunt.initConfig({
        /*
          Compile SASS code to CSS
         */
        sass: {
            options: {
                sourcemap: 'none',
                // style: 'expanded',
                outputStyle: 'compressed',
                noCache: true
            },
            dist: {
                files: [{
                    expand: true,
                    cwd: '',
                    src: [
                        'elements/**/*.scss'
                    ],
                    dest: '',
                    ext: '-min.css'
                }]
            }
        },
        /*
          Add css prefixes for compatibility (mainly for Firefox ESL)
         */
        postcss: {
            options: {
                map: false,
                processors: [
                    require('autoprefixer')({
                        browsers: ['firefox 24', 'IE 10', 'last 2 versions']
                    })
                ]
            },
            dist: {
                src: ['elements/**/*-min.css']
            }
        },
        /*
          Process JavaScript
         */
        uglify: {
```

```
            options: {
                preserveComments: false,
                srewIE8: true,
                sourceMap: false
            },
            /*
              Compile JavaScript on Polymer Elements
             */
            polymerjs: {
                options: {
                    mangle: false,
                    sourceMap: false
                },
                files: [{
                    expand: true,
                    src: ['elements/**/*.js'],
                    ext: '-min.js'
                }]
            }
        },

        inline: {

          dist: {
            files: [{
              expand: true,
              cwd: '',
              src: [
                  'elements/**/*.html'
              ],
              dest: '../../html/',
              ext: '.html'
            }]
          }
              },

        clean: {
            cssfiles: {
                options: {
                    'no-write': false
                },
                src: ["elements/**/*-min.css"]
            },
            jsfiles: {
                options: {
                    'no-write': false
                },
                src: ["elements/**/*-min.js"]
            }
        },

        /*
         * Make package documentation
         */
        execute: {
          target: {
            src: ['elements/makeIndex.js'],
            options: {
```

```
                cwd: "./elements/"
              }
            }
          }
    });

    grunt.loadNpmTasks('grunt-execute');
    grunt.loadNpmTasks('grunt-contrib-uglify');
    grunt.loadNpmTasks('grunt-postcss');
    grunt.loadNpmTasks('grunt-contrib-clean');
    grunt.loadNpmTasks('grunt-sass');
    grunt.loadNpmTasks('grunt-inline');

    grunt.registerTask('default', ['execute', 'sass', 'postcss', 'uglify', 'inline',
↪'clean']);
};
```

This specifies what Grunt has to do, where to find source files, and where to put built files.

Setting up documentation

Your cell will automatically generate documentation. So anyone running your cell can browse to <hostname>:<port>/<package-path>/html/index.html and explore what elements your cell contains.

Make the /src/html/elements/makeIndex.js file and edit the first few lines.

/src/html/elements/makeIndex.js

```
#!/usr/bin/env node
var repositoryPath = "https://svnweb.cern.ch/trac/cactus/browser/trunk/cactusprojects/
↪subsystem/supervisor/html-dev/elements/";
var projectName = "Subsystem Supervisor";
var projectPath = "subsystem/supervisor/html/elements/"


var fs = require('fs');
var path = require('path');

function getDirectories(srcpath) {
  return fs.readdirSync(srcpath).filter(function(file) {
    return fs.statSync(path.join(srcpath, file)).isDirectory() && file.indexOf('-') >␣
↪-1 && file.indexOf('template') == -1;
  });
}

var elements = getDirectories('.');
var result = [];
for (var i = 0; i < elements.length; i++) {
  var element = elements[i];
  var json = {name: element};
  if ( fs.existsSync(element + '/description.json') ) {
    var parsedJSON = require("./" + element + '/description.json');
    if (!parsedJSON.description) {
      console.error(element + '/description.json contains no description');
    }
    for (var property in parsedJSON) {
      if (parsedJSON.hasOwnProperty(property)) {
```

```
        json[property] = parsedJSON[property];
      }
    }
  } else {
    console.error(element + ' has no description.json file');
    json.description = "no description...";
  }
  if ( !fs.existsSync(element + '/index.html') ) {
    console.error(element + '/index.html does not exist');
  }
  result.push(json);
}

fs.readFile("./index_template.html", 'utf8', function(err, data) {
  if (err) {
    return console.error("index_template.html is missing or not readable");
  }
  data = data.replace(/<% data %>/g, JSON.stringify(result));
  data = data.replace(/<% repositoryPath %>/g, repositoryPath);
  data = data.replace(/<% projectName %>/g, projectName);
  data = data.replace(/<% projectPath %>/g, projectPath);
  fs.writeFile("index.html", data, function(err) {
      if(err) {
          return console.log(err);
      }

      console.log("index.html written");
  });
});
```

This will generate our package documentation using a template html file.

Create /src/html/elements/index_template.html

/src/html/elements/index_template.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title><% projectName %> documentation</title>

    <meta name="viewport" content="width=device-width, minimum-scale=1.0, initial-
↪scale=1, user-scalable=yes">
    <meta name="mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-capable" content="yes">

    <script>window.Polymer = window.Polymer || {};window.Polymer.dom = 'shadow';</
↪script>
    <script src="/extern/bower_components/webcomponentsjs/webcomponents-lite.js"></
↪script>

    <link rel="import" href="/extern/bower_components/paper-material/paper-material.
↪html">
    <link rel="import" href="/extern/bower_components/iron-icon/iron-icon.html">
    <link rel="import" href="/extern/bower_components/iron-icons/iron-icons.html">
    <link rel="import" href="/extern/bower_components/prism-element/prism-highlighter.
↪html">
    <link rel="import" href="/extern/bower_components/marked-element/marked-element.
↪html">
```

```
    <style>
      body {
        background-color: #FAFAFA;
      }
      paper-material {
        max-width: 800px;
        margin: auto;
        background-color: white;
      }
      marked-element {
        padding: 10px;
        padding-top: 0;
        display: block;
        margin-bottom: 1em;
      }
      div[title] {
        background-color: #81c784;
        color: black;
        font-family: roboto;
        font-weight: normal;
        margin: 0;
        padding: 10px;
        display: flex;
        flex-direction: row;
        padding: 15px 24px;
        font-size: .8em;
      }
      div[title] [name] {
        width: 250px;
      }
      h1 {
        color: black;
        font-family: roboto;
        font-weight: normal;
        margin: 0;
        padding: 15px 24px;
        font-size: 1.5em;
      }
      h2 {
        color: black;
        font-family: roboto;
        font-weight: normal;
        margin: 0;
        text-align: center;
        font-size: 1em;
        margin: 0;
        padding: 0;
        padding-left: 10px;
        padding-top: 10px;
      }
      a {
        display: flex;
        flex-direction: row;
        padding: 15px 24px;
        text-decoration: none;
        color: black;
        font-family: roboto;
        border-bottom: 1px solid #EFEFEF;
```

```
    }
    a div[options] {
      opacity: 0;
      transition: opacity 200ms;
    }
    a:hover {
      background-color: #fafafa;
    }
    a:hover div[options] {
      opacity: 1;
    }
    a div[name] {
      width: 250px;
      font-weight: bold;
    }
    a div[flex] {
      color: #757575;
    }
    [flex] {
      flex: 1;
    }
    a div {
      font-size: .8em;
    }
    iron-icon {
      height: 16px;
      z-index: 10;
      color: #797979;
    }
    iron-icon:hover {
      color: black;
    }
  </style>
</head>
<body>
  <h1><% projectName %> elements</h1>
  <paper-material elevation="1">
    <prism-highlighter></prism-highlighter>
    <h2>To use any of these elements in your project:</h2>
    <marked-element id="marked">
      <div class="markdown-html code" id="code"></div>
    </marked-element>
  </paper-material>


  <template is="dom-bind" id="app">
    <paper-material elevation="1">
      <div title>
        <div name>
          name
        </div>
        <div flex>
          description
        </div>
      </div>
      <template is="dom-repeat" items="{{data}}" as="element">
        <a href="[[element.name]]/index.html">
          <div name>
```

```
              [[element.name]]
            </div>
            <div flex>
              [[element.description]]
            </div>
            <div options>
              <iron-icon icon="description"></iron-icon>
              <iron-icon icon="code" on-click="showCode" onclick="return false;"></
↪iron-icon>
              <template is="dom-if" if="[[element.demo]]" restamp="true">
                <iron-icon icon="visibility" on-click="showDemo" onclick="return␣
↪false;"></iron-icon>
              </template>
            </div>
          </a>
        </template>
      </paper-material>
    </template>

    <script>
      var codeRepositoryRoot = "<% repositoryPath %>";
      window.addEventListener('WebComponentsReady', function(e) {
        var app = document.querySelector('#app');
        app.data = <% data %>;

        var el = document.getElementById('marked');
        var snippet = el.unindent('<link rel="import" href="<% projectPath %>element-
↪name/element-name.html">');
        // Boolean properties are displayed as checked="", so remove the ="" bit.
        snippet = snippet.replace(/=""/g, '');
        el.set('markdown', '```\n' + snippet + '\n' + '```');
      });
      app.showCode = function(e) {
        window.location = codeRepositoryRoot + e.model.element.name;
      }
      app.showDemo = function(e) {
        window.location = e.model.dataHost.dataHost.element.name + "/" + e.model.
↪dataHost.dataHost.element.demo;
      }
    </script>
  </body>
</html>
```

Now make a file html/index.html

/html/index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <meta http-equiv="refresh" content="0; URL='elements/index.html'" />
  </head>
  <body>

  </body>
</html>
```

This will redirect the user to the elements folder holding the documentation when they visit <hostname>:<port>/<package-name>/html/index.html

Using Grunt

Now you should be able to run

```
cd src/html
grunt
```

And you should see whatever elements are present in /src/html/elements are built and put into /html/elements (you probably don't have any elements now).

Your makefile will copy the /html folder into the RPM's. The src/html folder will not be copied and will not be present in production systems. Anything that does not need building can be safely copied into the /html folder. No build system will delete that folder.

Now that you have an update /html folder you can run

```
make rpm
```

The makefile will make a new rpm containing the updated /html folder.

Setting up a template

You will probably want to make some elements of your own now, but where to start? We'll give you a script that, using some template element, can generate a general element definition for you.

Create the file /src/html/elements/new-element.js file

/src/html/elements/new-element.js

```
#!/usr/bin/env node
process.stdin.resume();
process.stdin.setEncoding('utf8');
var util = require('util');
var ncp = require('ncp').ncp;
var replace = require("replace");
var renamer = require("renamer");
var path = require('path');
ncp.limit = 16;
var FindFiles = require("node-find-files");
var fs = require('fs');
var exec = require('child_process').exec;

process.stdout.write('name of the new element: ');
process.stdin.on('data', function (text) {

  var split = text.replace('\n', '').split('/');
  if (split.length == 1) {
    base = split[0];
    newname = base;
  } else if (split.length == 2) {
    base = split[0];
    newname = split[1];
  } else {
    console.error('\nname can only contain only one dash (/)');
```

```
    process.exit();
  }

  if (newname == '') {
    console.error('\nname cannot be empty');
    process.stdout.write('name of the new element: ');

  } else if (newname.indexOf('-') == -1) {
    console.error('\nname must contain a dash (-)');
    process.stdout.write('name of the new element: ');

  } else {
    if (split.length == 1 ) {
      console.log('creating new element <' + base + '>...');
    } else {
      console.log('creating new element <' + newname + '> in <' + base + '>...');
      return console.error("unfortunately we can't do this because we will mess up the .svn folders");
    }

    ncp('template-element', base, function (err) {
      if (err) { return console.error(err); }
      replace({
        regex: "template-element",
        replacement: newname,
        paths: [base],
        recursive: true,
        silent: true,
      });


      var finder = new FindFiles({
        rootFolder : base,
        filterFunction : function (path, stat) {
          return (path.indexOf('template-element') > -1) ? true : false;
        }
      });
      finder.on("match", function(strPath, stat) {
        // console.log(strPath + " -> " + strPath.replace('template-element', newname));
        fs.rename(strPath, strPath.replace('template-element', newname), function(err) {
          if ( err ) console.log('ERROR: ' + err);
        });
      })
      finder.on("complete", function() {
        console.log("removing any .svn folders in ", newname);
        exec('rm -rf `find ' + newname + ' -type d -name .svn`', function (err, stdout, stderr) {});
        console.log("Finished");
        process.exit();
      })
      finder.on("patherror", function(err, strPath) {
        // Note that an error in accessing a particular file does not stop the whole show
        console.log("Error for Path " + strPath + " " + err);
      })
      finder.on("error", function(err) {
```

```
        console.log("Global Error " + err);
        process.exit();
      })
      finder.startSearch();
    });
  }

});
```

This will copy a folder *template-element* and rename the appropriate code to the element name you specified.

Now create /src/html/elements/template-element/template-element.html

/src/html/elements/template-element/template-element.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/ts/common-elements/iron-flex-layout-attributes/iron-flex-
↪layout-attributes.html">

<!--
only the last comment before <dom-module ...> appears in the documentation
-->

<!--
Material design: [Click me](https://www.google.com/design/spec/components/text-fields.
↪html)

`template-element` is a template to be used when creating new elements.

Example:

    <template-element></template-element>
    <template-element test="test"></template-element>

### Styling
The following custom properties and mixins are available for styling:

Custom property | Description | Default
----------------|-------------|----------
`--my-custom-color` | A custom css property | `black`
`--my-mixin-name` | A custom mixin | `{}`

@demo demo/index.html


-->
<dom-module id="template-element">
  <template>
    <!-- this makes your element follow the general theme (things like fonts) -->
    <style include="reset-css"></style>

    <!-- this will allow you to use flexbox easily -->
    <!-- surf to /ts/common-elements/iron-flex-layout-attributes/index.html -->
    <style include="iron-flex-layout-attributes"></style>

    <link rel="stylesheet" type="text/css" href="css/template-element-min.css?__
↪inline=true">

```

```
    <h1>This is the template-element element!</h1>
    <paper-button raised on-click="makeDinosaur">[[someproperty]]</paper-button>

  </template>
  <script src="javascript/template-element-min.js?__inline=true"></script>
</dom-module>
```

Notice the big comment just before the *dom-module* line. This will be used to generate documentation for your element. Be sure to update the description of the element in this comment.

Now create /src/html/elements/template-element/description.json

/src/html/elements/template-element/description.json

```
{
  "description": "no description...",
  "demo": "demo/index.html"
}
```

This file gives a description for the package documentation that will be generated by Grunt. Change the description to something sensible when you generate a new element with new-element.js. delete the *demo* line if you, at one point, decide to not provide a demo.

Now create /src/html/elements/template-element/index.html

/src/html/elements/template-element/index.html

```
<!--
This file renders documentation of the element
-->
<!doctype html>
<html>
  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <script src="/extern/bower_components/webcomponentsjs/webcomponents-lite.js"></
↪script>
    <link rel="import" href="/extern/bower_components/iron-component-page/iron-
↪component-page.html">

  </head>
  <body unresolved>
    <!-- Note: if the main element for this repository doesn't
         match the folder name, add a src="<main-component>.html" attribute,
         where <main-component>.html" is a file that imports all of the
         components you want documented. -->
    <iron-component-page></iron-component-page>

  </body>
</html>
```

Now create /src/html/elements/template-element/javascript/template-element.js

/src/html/elements/template-element/javascript/template-element.js

```
Polymer({
    is: 'template-element',
```

```
3
4     behaviors: [
5       // Polymer.PaperInputBehavior
6     ],
7
8     properties: {
9       /**
10       * Fired when you make a dinosaur
11       *
12       * @event made-a-dinosaur
13       */
14
15      /**
16       * The message the element will show
17       */
18      someproperty: {
19        type: String,
20        value: "Hello, World!",
21        //alternatively, this can be a computed property, based on other properties
22        // computed: 'computeFullName(first, last)'
23        //someproperty-changed event will be fired when property changes (required␣
↪for data-binding to parent)
24        notify: true,
25        //element attribute will be updated when property changes
26        reflectToAttribute: true,
27        //function to execute if property changes
28        observer: '_disabledChanged',
29        //if true, cannot be updated except with _setSomeproperty(value)
30        readOnly: false
31      }
32    },
33    observers: [
34      // 'dosomething(someproperty, someotherproperty)'
35    ],
36
37    /**
38     * This will do something nice
39     *
40     * @param {Egg} egg The dinosaur egg.
41     * @return {Dinosaur}
42     */
43    makeDinosaur: function(egg) {
44      alert('you clicked the button!');
45      if (!egg) {egg = new Egg('velociraptor');}
46
47      // using this, developers can use your event to fire a function of their own
48      // <element-template on-made-a-dinosaur="customfunction"></element-template>
49      // the second argument is optional
50      this.fire('made-a-dinosaur', {fromEgg: egg});
51      return new Dinosaur(egg);
52    },
53
54    /**
55     * This is a private function, do not use
56     */
57    _destroyHumanity: function() {
58      // if you have a function you don't want others to use outside your element
59      // prefix the function with '_'
```

```
      dinosaurs = new Array();
      for (var i = 0; i < 100000000; i++) {
        dinosaurs[i] = new Dinosaur();
        dinosaurs[i]._killAllHumans();
      }
    },

    // Fires when an instance of the element is created
    // you have no data binding and the element does not contain html code yet
    created: function() {},

    // Fires when the local DOM has been fully prepared
    // data binding works and the template html is ready
    ready: function() {},

    // Fires when the element was inserted into the document
    attached: function() {},

    // Fires when the element was removed from the document
    detached: function() {},

    // Fires when an attribute was added, removed, or updated
    attributeChanged: function(name, type) {}
});
```

Note that this template serves as a boilerplate, and probably contains a lot of code you won't actually use. Delete lines you do not need in new elements you generate with this new-element.js script. Also notice the comments in the *properties* section and above every function definition. These are used to generate documentation for your element and follow the JSDoc syntax (http://usejsdoc.org/about-getting-started.html).

Now create /src/html/elements/template-element/demo/index.html

/src/html/elements/template-element/demo/index.html

```
<!doctype html>
<html>
  <head>
    <title>template-element demo</title>

    <meta name="viewport" content="width=device-width, minimum-scale=1.0, initial-
↪scale=1, user-scalable=yes">
    <meta name="mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-capable" content="yes">

    <script>window.Polymer = window.Polymer || {};window.Polymer.dom = 'shadow';</
↪script>
    <script src="/extern/bower_components/webcomponentsjs/webcomponents-lite.js"></
↪script>

    <link rel="import" href="../template-element.html">
  </head>
  <body unresolved>
    <template-element></template-element>
  </body>
</html>
```

This file is the demo. By default the demo only shows the element without any adjustments or data supplied to it. Adjust the demo if your element needs extra work or data before it becomes functional.

Now create /src/html/elements/template-element/css/template-element.scss

/src/html/elements/template-element/css/template-element.scss

```
// for more info about styling an element:
// https://www.polymer-project.org/1.0/docs/devguide/styling.html

:host {
  // always declare a display property for your element, otherwise it will apear
  // to have height and width = 0 but yet it renders content...
  display: block;
}
// :host can take an extra css selector as parameter
// this will apply when your element is used like this:
// <template-element disabled></template-element>
// or with data-binding
// <template-element disabled$="{{isDisabled}}"></template-element>
:host([disabled]) {
  color: gray;
}

.some-class, [some-attribute], some-element {
  // use custom css properties like this
  // color can be defined by another developer, it defaults to blue
  color: var(--my-custom-color, blue);
}
// another developer can do this now:
// template-element {
//   --my-custom-color: green;
// }

.some-class, [some-attribute], some-element {
  // use custom css mixins like this
  // another developer can now inject extra css at this point
  @apply(--my-mixin-name);
}
// another developer can do this now:
// template-element {
//   --my-mixin-name: #{'{
//     background-color: green;
//     border-radius: 4px;
//     border: 1px solid gray;
//   }'};
// }
```

In a generated element, you most probably won't need any of this code except the very first block (:host {display: block}). The rest serves as code examples. Notice that –my-custom-color and –my-mixin-name also appeared in the comment in template-element.html.

Now you should be able to run

```
cd src/html/elements
chmod +x new-element.js
./new-element.js
name of the new element: my-new-element
creating new element <my-new-element>...
removing any .svn folders in  my-new-element
Finished
```

And you will see a new folder my-new-element in /src/html/elements, ready for you to be developed further into whatever you want to build today.

Registering your elements in C++

When you open a web browser and navigate to your cell, your browser needs to be instructed to load your elements. AjaXell can do this for you, but you need to provide a list of elements.

Create a file /src/html/elements/elements.html Now, you don't have any elements yet. So this file will be empty for now. But here is an example how it would look like if you would have two elements *my-first-element* and *my-second-element*:

```
<link rel="import" href="my-first-element/my-first-element.html">
<link rel="import" href="my-second-element/my-second-element.html">
```

Now open the Cell.cc file and append the following line in the start of the Cell::init() function

```
void subsystemsupervisor::Cell::init()
{
  getContext()->addImport("/<package-path>/html/elements/elements.html");
  ...
```

<package-path> will depend on the name of your cell. For the subsystem supervisor the addImport line would be this

```
getContext()->addImport("/subsystem/supervisor/html/elements/elements.html");
```

1.2.2 The structure of a panel

A panel consists of C++ code rendering the data and one or more Polymer elements rendering the GUI.

A Polymer element consists of HTML, CSS, and JavaScript code. Each of these you can develop in a separate file.

C++

The main task of the C++ code is to provide the front-end code with data. This is something very important to realize. It will keep your code clean and easier to understand and change later on.

A minimal C++ panel looks like this:

```
#include "MyPanel.h"
#include "ajax/toolbox.h"
#include "ajax/PolymerElement.h"

#include "log4cplus/loggingmacros.h"
#include <iostream>

MyPanel::MyPanel( tsframework::CellAbstractContext* context, log4cplus::Logger&␣
↪logger ) : tsframework::CellPanel(context,logger) {
    logger_ = log4cplus::Logger::getInstance(logger.getName() +".MyPanel");
}
MyPanel::~MyPanel() {
    remove();
}

void MyPanel::layout(cgicc::Cgicc& cgi)
```

```
{
    remove();
    setEvent("user-clicked-button", ajax::Eventable::OnClick, this, &MyPanel::clicky);

    ajax::PolymerElement* mypanel = new ajax::PolymerElement("my-panel");
    add(mypanel);
}

void MyPanel::clicky(cgicc::Cgicc& cgi,std::ostream& out) {
    out << "This was executed because you clicked the button";
}
```

This code outputs '<my-panel></my-panel>' on page load. This is the name of our polymer element that renders the GUI for this panel.

It also registers a callback 'user-clicked-button'. When the server receives that callback it will execute clicky() and return whatever is piped into 'out'.

HTML

The main file of our Polymer element is the HTML file. It defines the visual structure and inserts our CSS and JavaScript code. It looks something like this:

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/extern/bower_components/paper-button/paper-button.html">
<!--
`<my-element>` is the Polymer element of the MyPanel panel.

It features a button the user can click. And when the user clicks this button
the server will say it clicked the button.

@authors me
-->
<dom-module id="my-element">
    <template>
        <link rel="import" type="css" href="css/my-element-min.css?__inline=true">
        <style include="reset-css"></style>

        <paper-button on-click="_doClickButton">Click me please</paper-button>
        <ts-ajax id="ajax"
                 data="{{ajax_result}}"
                 callback="user-clicked-button"
                 handle-as="text"></ts-ajax>
    </template>
    <script src="javascript/my-element-min.js?__inline=true"></script>
</dom-module>
```

JavaScript

The JavaScript of your element is what makes your element spring to life. It adds interactivity to your element. A basic JavaScript file looks like this:

```
Polymer({
  is: "my-element",
```

```
  properties: {
    ajax_result: {
      type: String,
      value: "you haven't clicked the button yet..."
    }
  },
  _doClickButton: function() {
    this.$.ajax.generateRequest();
  }
});
```

SASS

You may have heard about CSS, it allows you to style your HTML markup. It is very powerful. But it misses some features. One big missing features is the ability to nest your selectors. Or sometimes you want to create for-loops. Maybe you would like to set a variable for a color you use a lot...

This is where SASS comes in (http://sass-lang.com/). SASS is CSS with superpowers. You write your styles using SASS, and the Grunt build tool will translate it to normal CSS for you.

Also note that we use another tool called autoprefixer (https://css-tricks.com/autoprefixer/). This will allow you to not worry about using vendor-prefixes (for example -webkit-transition vs transition) to keep your CSS compatible with older browsers.

A minimal CSS file looks like this:

```
:host {
    display: block;
}
```

1.2.3 Demo 0: Hello World

Make the hello-world element

In your cell, run:

```
cd src/html/elements
./new-element.js
name of the new element: hello-world
creating new element <hello-world>...
removing any .svn folders in  hello-world
Finished
```

You now have a working *hello-world* element. We'll edit it soon.

Register the hello-world element

Edit src/html/elements/elements.html and add the following line

```
<link rel="import" href="hello-world/hello-world.html">
```

This will tell AjaXell to load our new element.

Edit the hello-world element

This is just a *hello, world!* example. We don't need much of the stuff our template generated.

Edit src/html/elements/hello-world/hello-world.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">

<!--
`hello-world` is the simplest panel imaginable. It displays 'Hello, World!',
nothing more.

Example:

    <hello-world></hello-world>

@demo demo/index.html
-->
<dom-module id="hello-world">
  <template>
    <style include="reset-css"></style>
    <link rel="stylesheet" type="text/css" href="css/hello-world-min.css?__inline=true
 ↪">

    <h1>Hello, World!</h1>

  </template>
  <script src="javascript/hello-world-min.js?__inline=true"></script>
</dom-module>
```

Note that we didn't delete the reset-css include. This is recommended to do, *reset-css* provides us with some general css (fonts, theme colors, ...).

Now edit src/html/elements/hello-world/css/hello-world.scss

```
:host {
  display: block;
}
```

It is recommended to always have a display directive in the :host{} section. This tells the browser how the element will behave inside a page. The most used are 'block', 'inline-block', and 'inline'. 'block' elements try to take as much width as possible, while 'inline' elements only take the width and height they need. 'inline-block' is an inline element that can still have a manually set width or height

Now edit src/html/elements/hello-world/javascript/hello-world.js

```
Polymer({
    is: 'hello-world'
});
```

This is the minimal required javascript for a Polymer element. It only declares the existence of the *hello-world* element.

Now execute Grunt to build our new Polymer element.

```
cd src/html
grunt
```

Make the hello-world panel

Make a new c++ file /src/common/panels/HelloWorld.cc

```
#include "subsystem/supervisor/panels/HelloWorld.h"
#include "ajax/PolymerElement.h"

using namespace subsystempanels;
HelloWorld::HelloWorld( tsframework::CellAbstractContext* context, log4cplus::Logger&␣
↪logger)
:tsframework::CellPanel(context,logger) {
  logger_ = log4cplus::Logger::getInstance(logger.getName() +".HelloWorld");
}

void HelloWorld::layout(cgicc::Cgicc& cgi) {
  remove();
  add(new ajax::PolymerElement("hello-world"));
}
```

The remove(); function clears any previously existing output buffer from the HelloWorld panel. If you remove that line and you request the panel twice, you get 2 *hello-world* elements.

Make the include/subsystem/supervisor/panels/HelloWorld.h file.

```
#ifndef _subsystem_supervisor_panels_HelloWorld_h_
#define _subsystem_supervisor_panels_HelloWorld_h_

#include "ts/framework/CellPanel.h"
#include "log4cplus/logger.h"
#include "cgicc/Cgicc.h"

namespace subsystempanels {
  class HelloWorld: public tsframework::CellPanel {
    public:
      HelloWorld(tsframework::CellAbstractContext* context, log4cplus::Logger&␣
↪logger);
      void layout(cgicc::Cgicc& cgi);
  };
}
#endif
```

Register the new class in the Makefile.

```
Sources=\
    version.cc\
    Cell.cc\
    CellContext.cc\
    Configuration.cc\
    ...
    panels/HelloWorld.cc \
    ...
```

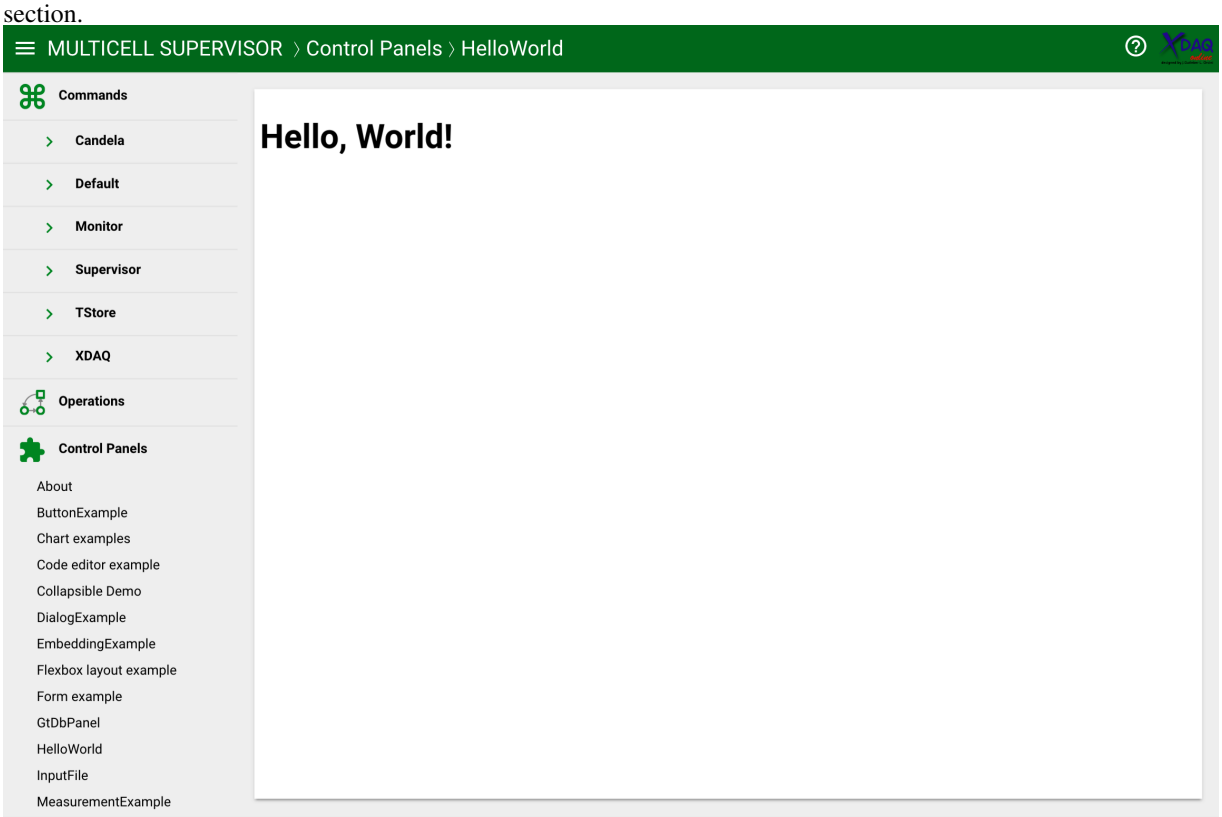
Now register your new panel in the menu so users can access it.

Edit src/common/Cell.cc

```
#include "subsystem/supervisor/panels/HelloWorld.h"
...
void subsystemsupervisor::Cell::init() {
```

```
  ...
  tsframework::CellPanelFactory* panelF = getContext()->getPanelFactory();
  ...
  panelF->add<subsystempanels::HelloWorld>("HelloWorld");
```

Now you can compile your cell and you should see the HelloWorld panel in the menu under the 'control-panels' section.



Also your element has created some documentation. Surf to <hostname>:<port>/<package-name>/html/index.html and you will see the package documentation for your cell. *hello-world* will be in there, and clicking it brings up the documentation for your *hello-world* element.

1.2.4 Demo 1: Ajax and data binding

Probably you would like your C++ code to supply some data to your panel. We will use the *ts-ajax* element in the *common-elements* package to retrieve our data, then we will use data-binding to display the data in our panel.

Make the data-binding element

In your cell, run:

```
cd src/html/elements
./new-element.js
name of the new element: data-binding
creating new element <data-binding>...
removing any .svn folders in  data-binding
Finished
```

Register the data-binding element

Edit src/html/elements/elements.html and add the following line

```
<link rel="import" href="data-binding/data-binding.html">
```

This will tell AjaXell to load our new element.

Edit the data-binding element

Edit src/html/elements/data-binding/data-binding.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/ts/common-elements/ts-ajax/ts-ajax.html">
<link rel="import" href="/extern/bower_components/paper-button/paper-button.html">

<!--
`data-binding` retrieves some data from C++ and displays it to the user

Example:

    <data-binding></data-binding>

-->
<dom-module id="data-binding">
  <template>
    <style include="reset-css"></style>
    <link rel="stylesheet" type="text/css" href="css/data-binding-min.css?__
↪inline=true">

    <p>This is the data-binding example. It will fetch some data using the
    `ts-ajax` element and display it here.</p>

    <h1>example 1</h1>
    <ts-ajax data="{{example1}}" callback="example1function" handle-as="text" auto></
↪ts-ajax>
    <span>[[example1]]</span>

    <h1>example 2</h1>
    <ts-ajax id="example2" data="{{example2}}" callback="example2function" handle-as=
↪"json"></ts-ajax>
    <paper-button raised on-click="doCallback">Do example2function callback</paper-
↪button><br>

    <template is="dom-repeat" items="[[example2]]" as="item">
      <span>[[item]]</span><br>
    </template>

  </template>
  <script src="javascript/data-binding-min.js?__inline=true"></script>
</dom-module>
```

Note the {{...}} and [[...]] code. This is our data binding code. Consider the highlighted lines. Line 23 tells Polymer to link the *data* variable from ts-ajax with our own *example1* variable. This way, if ts-ajax changes its *data* variable our own *example1* variable will change too.

When we use the {{...}} syntax this change goes both ways. [[...]] goes one way only. Use the latter to display some final result as we did in line 24, where we don't anticipate a source of change.

Now edit src/html/elements/data-binding/css/data-binding.scss

```
:host {
  display: block;
}
```

Now edit src/html/elements/data-binding/javascript/data-binding.js

```
Polymer({
    is: 'data-binding',

    properties: {
      example1: {
        type: String,
        value: "no data from C++ yet..."
      },
      example2: {
        type: Array,
        value: function() {
          return ["no data from C++ yet..."];
        }
      }
    },

    doCallback: function() {
      // this.$ is a shorthand selector,
      // it allows us to select an element in our template by id
      this.$.example2.generateRequest();
    }
});
```

Now execute Grunt to build our new Polymer element.

```
cd src/html
grunt
```

Make the data-binding panel

Make a new c++ file /src/common/panels/DataBinding.cc

```
#include "subsystem/supervisor/panels/DataBinding.h"
#include "ajax/PolymerElement.h"
#include "json/json.h"
#include <sstream>

using namespace subsystempanels;
DataBinding::DataBinding( tsframework::CellAbstractContext* context, log4cplus::
↪Logger& logger)
:tsframework::CellPanel(context,logger) {
  logger_ = log4cplus::Logger::getInstance(logger.getName() +".DataBinding");
}

void DataBinding::layout(cgicc::Cgicc& cgi) {
  remove();
```

```
  setEvent("example1function", ajax::Eventable::OnClick, this, &DataBinding::
↪example1);
  setEvent("example2function", ajax::Eventable::OnClick, this, &DataBinding::
↪example2);
  add(new ajax::PolymerElement("data-binding"));
}

void DataBinding::example1(cgicc::Cgicc& cgi,std::ostream& out) {
  out << "This text is generated using C++!";
}

void DataBinding::example2(cgicc::Cgicc& cgi,std::ostream& out) {
  Json::Value root(Json::arrayValue);
  for (size_t i = 0; i < 10; i++) {
    std::stringstream ss;
    ss << "This is text " << i << " generated by C++";
    root.append(ss.str());
  }
  out << root;
}
```

Notice that in the HTML code earlier we specified callback=”example1function” in one of the ts-ajax elements. Notice the #include “json/json.h” line. We import the jsoncpp library this way in order to create an array of strings in example2().

Edit your Makefile to add jsoncpp as a dependency.

```
DependentLibraries = tsframework tsajaxell ... jsoncpp
```

Make the include/subsystem/supervisor/panels/DataBinding.h file.

```
#ifndef _subsystem_supervisor_panels_DataBinding_h_
#define _subsystem_supervisor_panels_DataBinding_h_

#include "ts/framework/CellPanel.h"
#include "log4cplus/logger.h"
#include "cgicc/Cgicc.h"

namespace subsystempanels {
  class DataBinding: public tsframework::CellPanel {
    public:
      DataBinding(tsframework::CellAbstractContext* context, log4cplus::Logger&␣
↪logger);
      void layout(cgicc::Cgicc& cgi);
    private:
     void example1(cgicc::Cgicc& cgi,std::ostream& out);
     void example2(cgicc::Cgicc& cgi,std::ostream& out);
  };
}
#endif
```

Register the new class in the Makefile.

```
Sources=\
    version.cc\
    Cell.cc\
    CellContext.cc\
    Configuration.cc\
```

```
    ...
    panels/DataBinding.cc \
    ...
```

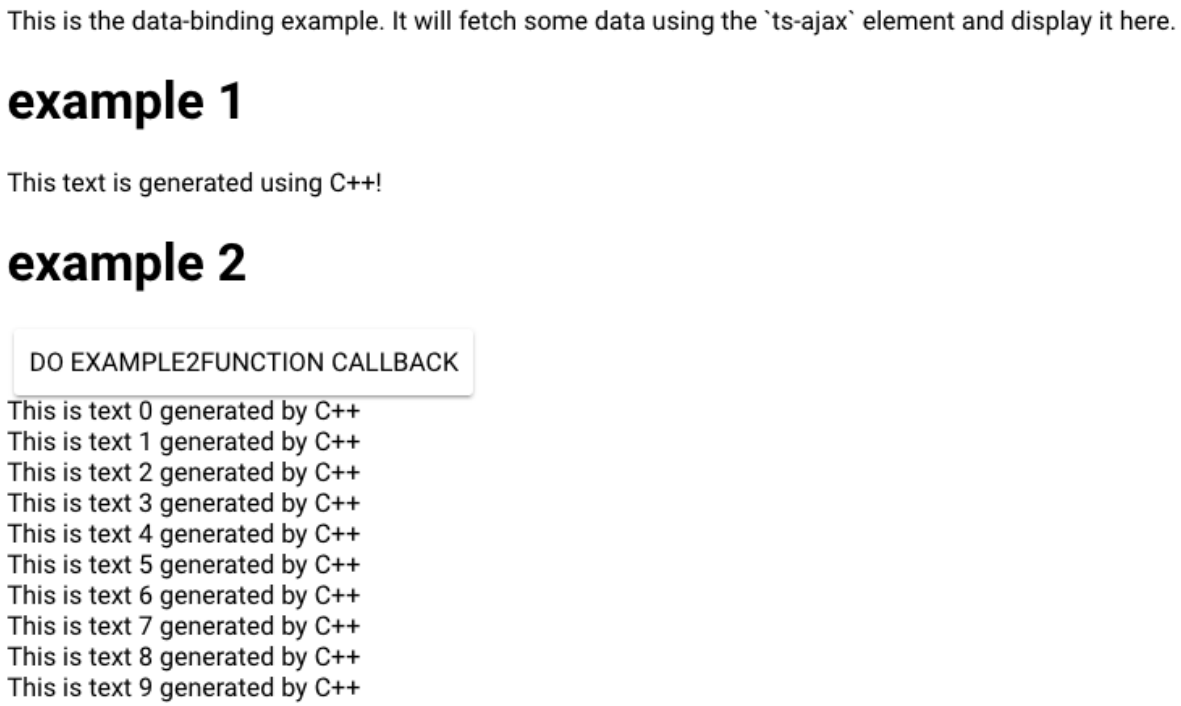
Now register your new panel in the menu so users can access it.

Edit src/common/Cell.cc

```
#include "subsystem/supervisor/panels/DataBinding.h"
...
void subsystemsupervisor::Cell::init() {
  ...
  tsframework::CellPanelFactory* panelF = getContext()->getPanelFactory();
  ...
  panelF->add<subsystempanels::DataBinding>("ts-ajax and data-binding");
```

That's right, you can have spaces in your menu names.

Now you can compile your cell and you should see the "ts-ajax and data-binding" panel in the menu under the 'control-panels' section.



Also your element has created some documentation. Surf to <hostname>:<port>/<package-name>/html/index.html and you will see the package documentation for your cell. *data-binding* will be in there, and clicking it brings up the documentation for your *data-binding* element.

Be sure to check out the documentation for the *ts-ajax* element at <hostname>:<port>/ts/common-elements/html/index.html

1.2.5 Demo 2: The flexbox layout

To setup a layout of your panels you can use the flexbox layout system. This is a set of new CSS directives that seeks to make designing layouts much simpler. It would deprecate the use of *float: left* and other nonsense.

Make the flexbox-layout element

In your cell, run:

```
cd src/html/elements
./new-element.js
name of the new element: flexbox-layout
creating new element <flexbox-layout>...
removing any .svn folders in  flexbox-layout
Finished
```

Register the flexbox-layout element

Edit src/html/elements/elements.html and add the following line

```
<link rel="import" href="flexbox-layout/flexbox-layout.html">
```

This will tell AjaXell to load our new element.

Edit the flexbox-layout element

Edit src/html/elements/flexbox-layout/flexbox-layout.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/ts/common-elements/ts-colors/ts-colors.html">
<link rel="import" href="/ts/common-elements/iron-flex-layout-attributes/iron-flex-
↪layout-attributes.html">

<!--
`flexbox-layout` is a demo showcasing the capabilities of iron-flex-layout-attributes.

@demo demo/index.html

-->
<dom-module id="flexbox-layout">
  <template>
    <style include="reset-css"></style>
    <style include="ts-colors"></style>

    <!-- this will allow you to use flexbox easily -->
    <!-- surf to /ts/common-elements/iron-flex-layout-attributes/index.html -->
    <style include="iron-flex-layout-attributes"></style>

    <link rel="stylesheet" type="text/css" href="css/flexbox-layout-min.css?__
↪inline=true">
  </template>
  <script src="javascript/flexbox-layout-min.js?__inline=true"></script>
</dom-module>
```

We'll add more stuff as we go along. Note the import for *ts-colors*. It will allow us to do easily add colors to our stuff. It implements the material design color pallete (https://www.google.com/design/spec/style/color.html#color-color-palette).

The following code will contain a blue box:

```
<div blue-400>this will have a blue background</div>
<div blue-100>this will have a light blue background</div>
```

Now edit src/html/elements/flexbox-layout/css/flexbox-layout.scss

```
:host {
  display: block;
}

[square] {
  width: 100px;
  height: 100px;
  margin: 1em;
  img {
    width: 100px;
    height: 100px;
  }
}
```

Now edit src/html/elements/flexbox-layout/javascript/flexbox-layout.js

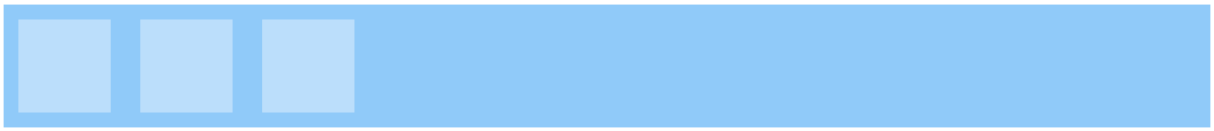
```
Polymer({
    is: "flexbox-layoutexample"
});
```

Flexbox layouts

Horizontal layout Add the following code to the HTML template

```
<h2>Horizontal layout</h2>
<div horizontal layout blue-200>
  <div square blue-100></div>
  <div square blue-100></div>
  <div square blue-100></div>
</div>
```

It will render this:



Horizontal layout with flex The flex attribute instructs an element to take as much space as possible in the direction of the layout (horizontal or vertical).

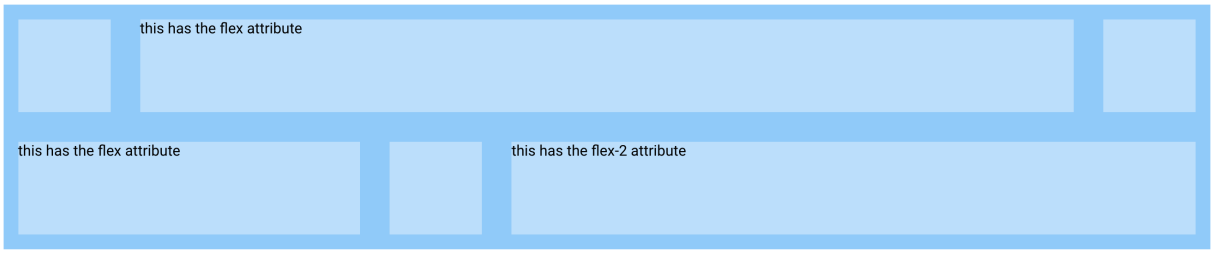
If there are multiple elements with the flex attribute the available space will be divided equally between them.

There are also the flex-2, flex-3, ..., flex-12 attributes. When multiple elements in the same layout have flex attributes, this will assign a greater weight to the elements. An element with the flex-2 attribute will be twice as big as the element with the flex attribute in the same layout.

Add the following code to the HTML template

```
<div horizontal layout blue-200>
  <div square blue-100></div>
  <div square blue-100 flex>this has the flex attribute</div>
  <div square blue-100></div>
</div>
<div horizontal layout blue-200>
  <div square blue-100 flex>this has the flex attribute</div>
  <div square blue-100></div>
  <div square blue-100 flex-2>this has the flex-2 attribute</div>
</div>
```

It will render this:

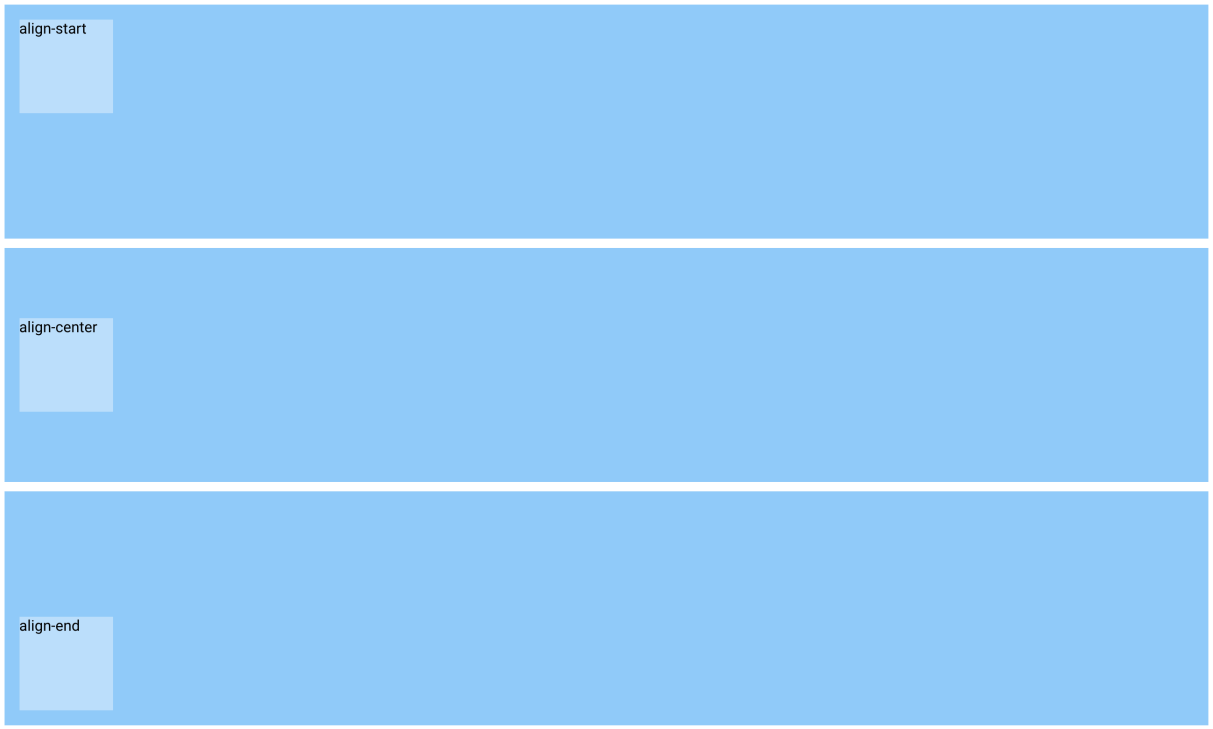


vertical alignment When you have for example a horizontal layout you might like to also control how the elements in the layout behave vertically. Same goes the other way, you might like to control horizontal behavior of elements in a vertical layout.

Add the following code to the HTML template

```
<div horizontal layout blue-200 align-start style="height: 250px;margin: 10px;">
  <div square blue-100>align-start</div>
</div>
<div horizontal layout blue-200 align-center style="height: 250px;margin: 10px;">
  <div square blue-100>align-center</div>
</div>
<div horizontal layout blue-200 align-end style="height: 250px;margin: 10px;">
  <div square blue-100>align-end</div>
</div>
```

It will render this:



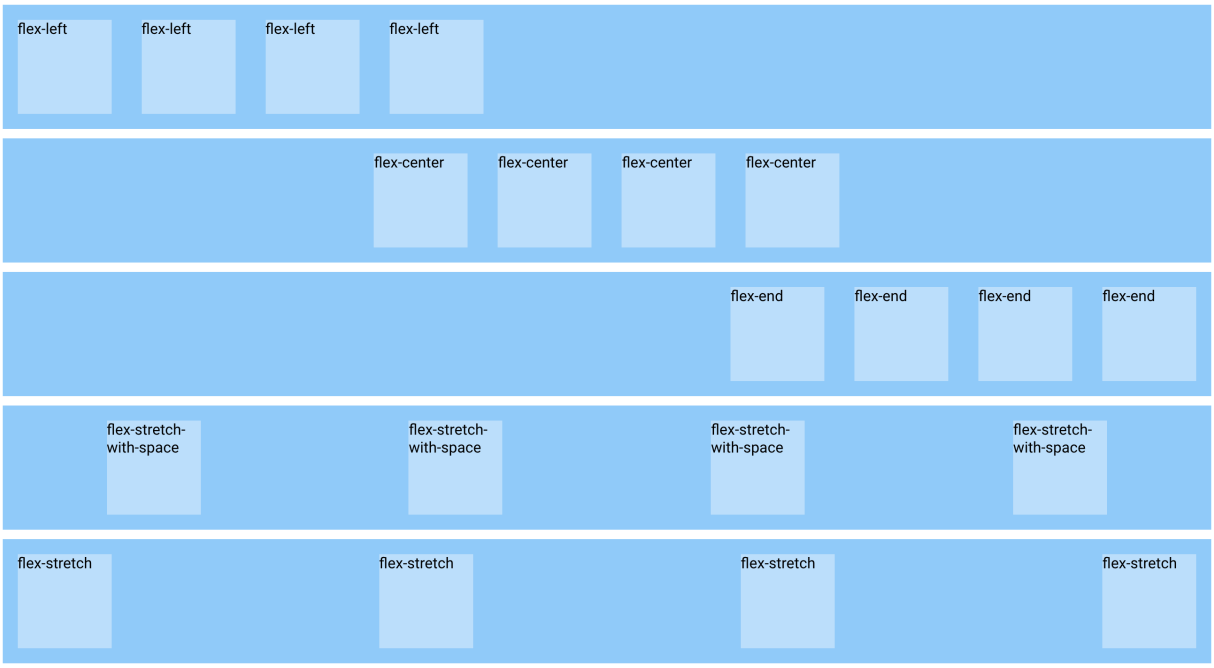
flex alignment You can also specify options for behavior of elements in the layout direction. For example you might want to horizontally center a set of horizontally aligned elements.

Add the following code to the HTML template

```
<div horizontal layout blue-200 flex-left style="margin: 10px;">
  <div square blue-100>flex-left</div>
  <div square blue-100>flex-left</div>
  <div square blue-100>flex-left</div>
  <div square blue-100>flex-left</div>
</div>
<div horizontal layout blue-200 flex-center style="margin: 10px;">
  <div square blue-100>flex-center</div>
  <div square blue-100>flex-center</div>
  <div square blue-100>flex-center</div>
  <div square blue-100>flex-center</div>
</div>
<div horizontal layout blue-200 flex-end style="margin: 10px;">
  <div square blue-100>flex-end</div>
  <div square blue-100>flex-end</div>
  <div square blue-100>flex-end</div>
  <div square blue-100>flex-end</div>
</div>
<div horizontal layout blue-200 flex-stretch-with-space style="margin: 10px;">
  <div square blue-100>flex-stretch-with-space</div>
  <div square blue-100>flex-stretch-with-space</div>
  <div square blue-100>flex-stretch-with-space</div>
  <div square blue-100>flex-stretch-with-space</div>
</div>
<div horizontal layout blue-200 flex-stretch style="margin: 10px;">
  <div square blue-100>flex-stretch</div>
```

```
    <div square blue-100>flex-stretch</div>
    <div square blue-100>flex-stretch</div>
    <div square blue-100>flex-stretch</div>
  </div>
```

It will render this:



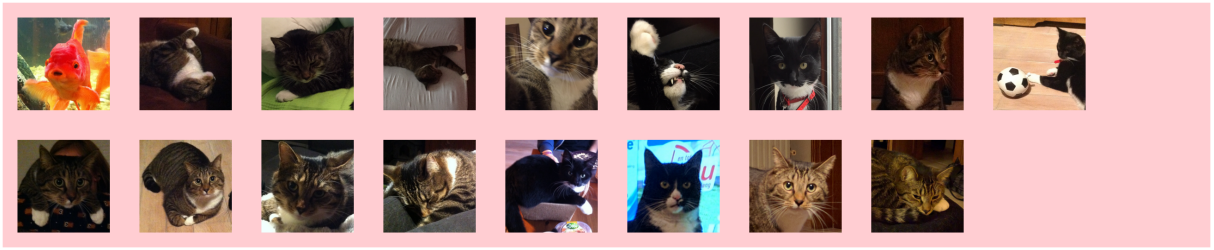
wrapped content blocks

You can instruct a layout to break its set of elements into multiple lines.

Add the following code to the HTML template

```
<div horizontal layout wrap red-100>
  <div square red-300><img src="../../pics/fish.png" /></div>
  <div square red-300><img src="../../pics/cat1.png" /></div>
  <div square red-300><img src="../../pics/cat2.png" /></div>
  <div square red-300><img src="../../pics/cat3.png" /></div>
  <div square red-300><img src="../../pics/cat4.png" /></div>
  <div square red-300><img src="../../pics/cat5.png" /></div>
  <div square red-300><img src="../../pics/cat6.png" /></div>
  <div square red-300><img src="../../pics/cat7.png" /></div>
  <div square red-300><img src="../../pics/cat8.png" /></div>
  <div square red-300><img src="../../pics/cat9.png" /></div>
  <div square red-300><img src="../../pics/cat10.png" /></div>
  <div square red-300><img src="../../pics/cat11.png" /></div>
  <div square red-300><img src="../../pics/cat12.png" /></div>
  <div square red-300><img src="../../pics/cat13.png" /></div>
  <div square red-300><img src="../../pics/cat14.png" /></div>
  <div square red-300><img src="../../pics/cat15.png" /></div>
  <div square red-300><img src="../../pics/cat16.png" /></div>
</div>
```

It will render this:

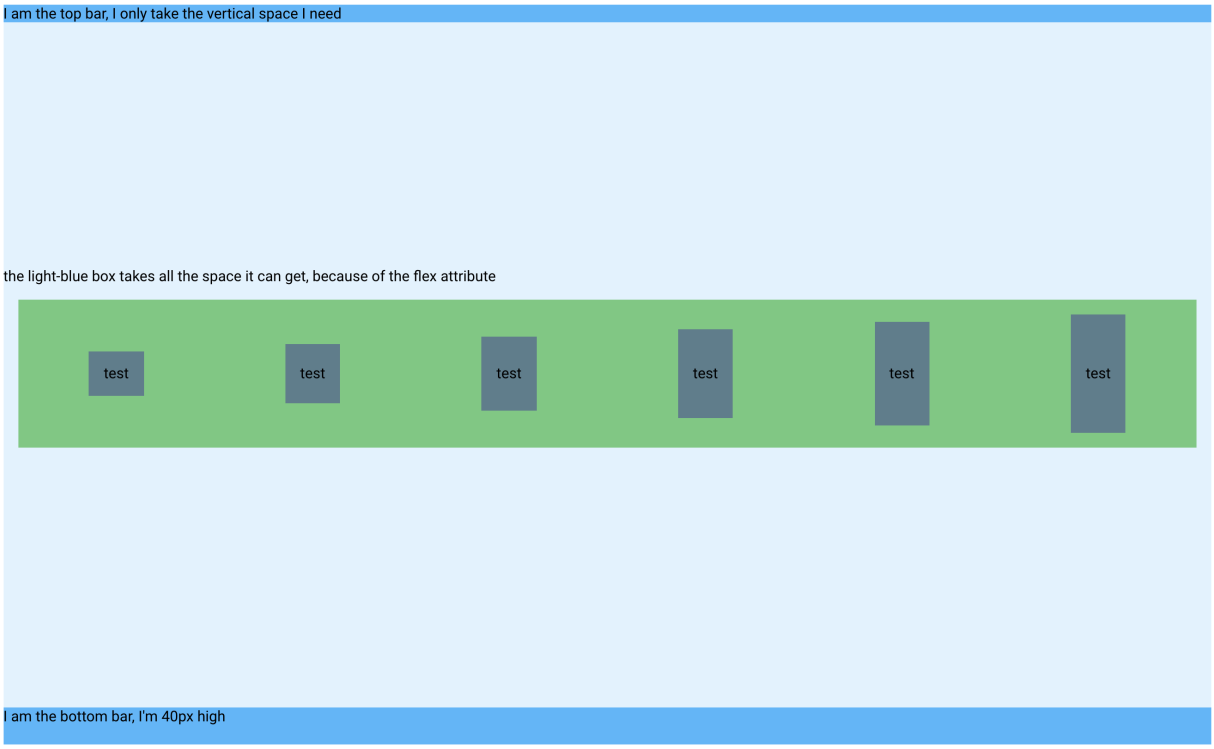


header and footer, vertically centered content

This layout generates a header and a footer, the content has the flex attribute and takes the rest of the vertical space.

```
<div vertical layout style="height:800px;">
  <div blue-300>I am the top bar, I only take the vertical space I need</div>
  <div blue-50 flex vertical layout flex-center>
    <span>the light-blue box takes all the space it can get, because of the flex
↪attribute</span>

    <div horizontal layout
         green-300
         flex-stretch-with-space
         align-center
         style="height: 10em;margin: 1em;">
      <span blue-grey-500 style="margin:1em;padding:1em;line-height:1em;">test</span>
      <span blue-grey-500 style="margin:1em;padding:1em;line-height:2em;">test</span>
      <span blue-grey-500 style="margin:1em;padding:1em;line-height:3em;">test</span>
      <span blue-grey-500 style="margin:1em;padding:1em;line-height:4em;">test</span>
      <span blue-grey-500 style="margin:1em;padding:1em;line-height:5em;">test</span>
      <span blue-grey-500 style="margin:1em;padding:1em;line-height:6em;">test</span>
    </div>

  </div>
  <div blue-300 style="height: 40px;">I am the bottom bar, I'm 40px high</div>
</div>
```

It will render this:



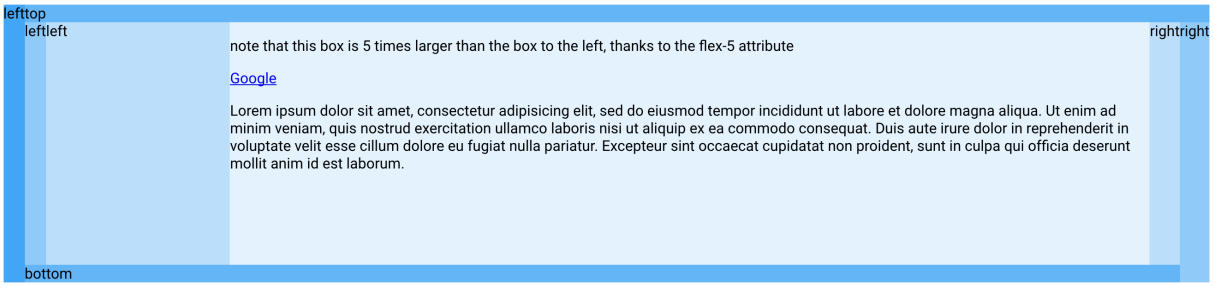
Impossibly complex layout 1

This code:

```
<!-- big box, children will be put next to each other -->
<div horizontal layout style="height: 300px;">
    <div blue-400>left</div>

    <!-- second big box, children will be put on top of each other -->
    <div vertical layout flex>
        <div blue-300>top</div>

        <div horizontal layout flex>

            <div vertical layout flex>

                <div horizontal layout flex>
                    <div blue-200>left</div>
                    <div blue-100 flex>left</div>
                    <div blue-50 flex-5>
                        <p>note that this box is 5 times larger than the box to the left, thanks to the flex-5 attribute</p>
                        <p><a href="https://www.google.be">Google</a></p>
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
                    </div>
```

```
                    <div blue-100>right</div>
                </div>

                <div blue-300>bottom</div>
            </div>

            <div blue-200>right</div>
        </div>

    </div>

</div>
```

Will render this:

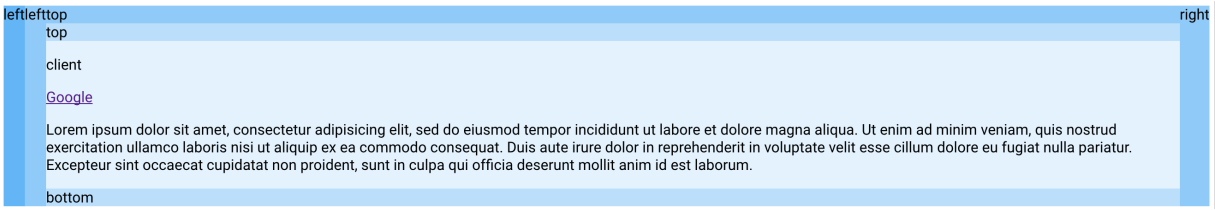


Impossibly complex layout 2

This code:

```
<div horizontal layout>
    <div blue-300>left</div>
    <div blue-200>left</div>
    <div vertical layout>
        <div blue-200>top</div>
        <div blue-100>top</div>
        <div blue-50>
            <p>client</p>
            <p><a href="">Google</a></p>
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do␣
↪eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim␣
↪veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo␣
↪consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum␣
↪dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident,␣
↪sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
        </div>
        <div blue-100>bottom</div>
    </div>
    <div blue-200>right</div>
</div>
```

Will render this:



Now execute Grunt to build our new Polymer element.

```
cd src/html
grunt
```

Make the flexbox-layout panel

Make a new c++ file /src/common/panels/FlexboxLayout.cc

```
#include "subsystem/supervisor/panels/FlexboxLayout.h"
#include "ajax/PolymerElement.h"

using namespace subsystempanels;
FlexboxLayout::FlexboxLayout( tsframework::CellAbstractContext* context, log4cplus::
 ↪Logger& logger)
:tsframework::CellPanel(context,logger) {
  logger_ = log4cplus::Logger::getInstance(logger.getName() +".FlexboxLayout");
}

void FlexboxLayout::layout(cgicc::Cgicc& cgi) {
  remove();
  add(new ajax::PolymerElement("flexbox-layout"));
}
```

Make the include/subsystem/supervisor/panels/FlexboxLayout.h file.

```
#ifndef _subsystem_supervisor_panels_FlexboxLayout_h_
#define _subsystem_supervisor_panels_FlexboxLayout_h_

#include "ts/framework/CellPanel.h"
#include "log4cplus/logger.h"
#include "cgicc/Cgicc.h"

namespace subsystempanels {
  class FlexboxLayout: public tsframework::CellPanel {
    public:
      FlexboxLayout(tsframework::CellAbstractContext* context, log4cplus::Logger&␣
 ↪logger);
      void layout(cgicc::Cgicc& cgi);
  };
}
#endif
```

Register the new class in the Makefile.

```
Sources=\
    version.cc\
    Cell.cc\
    CellContext.cc\
```

```
    Configuration.cc\
    ...
    panels/FlexboxLayout.cc \
    ...
```

Now register your new panel in the menu so users can access it.

Edit src/common/Cell.cc

```
#include "subsystem/supervisor/panels/FlexboxLayout.h"
...
void subsystemsupervisor::Cell::init() {
  ...
  tsframework::CellPanelFactory* panelF = getContext()->getPanelFactory();
  ...
  panelF->add<subsystempanels::FlexboxLayout>("Flexbox layout examples");
```

Now you can compile your cell and you should see the "Flexbox layout examples" panel in the menu under the 'control-panels' section.

Also your element has created some documentation. Surf to <hostname>:<port>/<package-name>/html/index.html and you will see the package documentation for your cell. *flexbox-layout* will be in there, and clicking it brings up the documentation for your *flexbox-layout* element.

1.2.6 Demo 3: Sending data to the server

The *ts-ajax* element can also be used to send data back to the server. To demonstrate this we'll build a simple form.

Make the form-example element

In your cell, run:

```
cd src/html/elements
./new-element.js
name of the new element: form-example
creating new element <form-example>...
removing any .svn folders in  form-example
Finished
```

Register the form-example element

Edit src/html/elements/elements.html and add the following line

```
<link rel="import" href="form-example/form-example.html">
```

This will tell AjaXell to load our new element.

Edit the form-example element

Edit src/html/elements/form-example/form-example.html

/src/html/elements/form-example/form-example.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/ts/common-elements/iron-flex-layout-attributes/iron-flex-layout-attributes.html">
<link rel="import" href="/extern/bower_components/paper-material/paper-material.html">
<link rel="import" href="/extern/bower_components/paper-input/paper-input.html">
<link rel="import" href="/extern/bower_components/paper-input/paper-textarea.html">
<link rel="import" href="/extern/bower_components/iron-icon/iron-icon.html">
<link rel="import" href="/extern/bower_components/iron-icons/iron-icons.html">
<link rel="import" href="/extern/bower_components/paper-checkbox/paper-checkbox.html">
<link rel="import" href="/extern/bower_components/paper-radio-button/paper-radio-button.html">
<link rel="import" href="/extern/bower_components/paper-radio-group/paper-radio-group.html">
<link rel="import" href="/extern/bower_components/paper-slider/paper-slider.html">
<link rel="import" href="/extern/bower_components/paper-fab/paper-fab.html">
<link rel="import" href="/extern/bower_components/paper-toggle-button/paper-toggle-button.html">
<link rel="import" href="/ts/common-elements/ts-ajax/ts-ajax.html">
<link rel="import" href="complex-input.html">

<!-- for animations -->
<link rel="import" href="/extern/bower_components/neon-animation/neon-shared-element-animatable-behavior.html">
<link rel="import" href="/extern/bower_components/neon-animation/neon-animatable-behavior.html">
<link rel="import" href="/extern/bower_components/neon-animation/neon-animation-runner-behavior.html">
<link rel="import" href="/extern/bower_components/neon-animation/animations/fade-in-animation.html">
<link rel="import" href="/extern/bower_components/neon-animation/animations/cascaded-animation.html">
<link rel="import" href="/extern/bower_components/neon-animation/animations/transform-animation.html">

<!--
`<polymer-formexample>` is an element that demonstrates the most commong things found
in a form made in a polymer fashion.

It uses flexbox to setup the layout. This layout is set via attribute selectors.
with a media querie these attributes can be changed, and so achieve a responsive layout

data binding is used to:
 - bind the media query and the attributes
 - bind some components together to make the form interactive
    example: the checkboxes aren't visible if the toggle-button isn't checked

@authors Glenn Dirkx
-->
<dom-module id="form-example">
  <template>
    <style include="reset-css"></style>
    <style include="iron-flex-layout-attributes"></style>

    <link rel="stylesheet" type="text/css" href="css/form-example-min.css?__inline=true">

```

```
47      <ts-ajax id="ajax"
48               callback="submit"
49               data="{{response}}"
50               parameters='["sometextinput", "defaultvaluetext", "password", "textarea
    ↪", "charcounter", "charcounter10", "letters", "letters2", "username", "ssn",
    ↪"likespizza", "withcheese", "withsalami", "withpineapple", "withonion", "withkebab
    ↪", "favoritetvstuff", "slider1", "slider2", "slider3", "slider4"]'
51               sometextinput={{text1}}
52               defaultvaluetext="{{text2}}"
53               password="{{text3}}"
54               textarea="{{text4}}"
55               charcounter="{{text6}}"
56               charcounter10="{{text7}}"
57               letters="{{text8}}"
58               letters2="{{text9}}"
59               username="{{text10}}"
60               ssn="{{text11}}"
61               likespizza="{{likesPizza}}"
62               withcheese="{{withCheese}}"
63               withsalami="{{withSalami}}"
64               withpineapple="{{withPineapple}}"
65               withonion="{{withOnion}}"
66               withkebab="{{withKebab}}"
67               favoritetvstuff="{{favoriteTVstuff}}"
68               slider1="{{slider1}}"
69               slider2="{{slider2}}"
70               slider3="{{slider3}}"
71               slider4="{{slider4}}"></ts-ajax>
72               <!-- notice the single & double quotes in the parameters variable
73               do yourself a favor and do not use capital letters in parameter names -->
74
75      <section horizontal layout>
76
77        <paper-material elevation="1" flex>
78          <paper-input label="some text input"
79                       value="{{text1}}"></paper-input>
80
81          <paper-input label="text input with default value"
82                       value="{{text2}}"></paper-input>
83
84          <paper-input label="type your current CERN password here"
85                       type="password"
86                       value="{{text3}}"></paper-input>
87
88          <paper-textarea label="this is actually a text area, it will grow as needed"
89                          value="{{text4}}"></paper-textarea>
90
91          <paper-input label="this one is disabled"
92                       disabled
93                       value="{{text5}}"></paper-input>
94
95          <paper-input label="simple character counter"
96                       char-counter
97                       value="{{text6}}"></paper-input>
98
99          <paper-input label="input with at most 10 characters"
100                      char-counter
101                      maxlength="10"
```

```
                    value="{{text7}}"></paper-input>

        <paper-input label="this input requires letters only"
                    auto-validate pattern="[a-zA-Z]*"
                    error-message="letters only!"
                    value="{{text8}}"></paper-input>

        <paper-input label="this input will only let you type letters"
                    auto-validate
                    allowed-pattern="[a-zA-Z]"
                    value="{{text9}}"></paper-input>

        <paper-input label="username"
                    value="{{text10}}">
          <iron-icon icon="mail" prefix></iron-icon>
          <div suffix>@email.com</div>
        </paper-input>

        <paper-input-container always-float-label auto-validate attr-for-value="value
↪">
          <label>Social Security Number</label>
          <complex-input class="paper-input-input" value="{{text11}}"></complex-input>
          <paper-input-error>SSN invalid!</paper-input-error>
        </paper-input-container>

      </paper-material>

      <div vertical layout flex>

        <paper-material elevation="1" flex vertical layout>
          <paper-toggle-button checked="{{likesPizza}}">I like pizza</paper-toggle-
↪button>
          <template is="dom-if" if="{{likesPizza}}">
            <paper-checkbox checked="{{withCheese}}">with cheese</paper-checkbox>
            <paper-checkbox checked="{{withSalami}}">with salami</paper-checkbox>
            <paper-checkbox checked="{{withPineapple}}">with pineapple</paper-
↪checkbox>
            <template is="dom-if" if="{{withPineapple}}" restamp="true">
              <p pineapple-note>You have terrible taste</p>
            </template>
            <paper-checkbox checked="{{withOnion}}">with onion</paper-checkbox>
            <paper-checkbox checked="{{withKebab}}">with kebab</paper-checkbox>
          </template>
        </paper-material>

        <paper-material elevation="1">
          <paper-radio-group selected="{{favoriteTVstuff}}" vertical layout>
            <paper-radio-button name="starwars">Star Wars</paper-radio-button>
            <paper-radio-button name="startrek">Star Trek</paper-radio-button>
            <paper-radio-button name="drwho">Dr Who</paper-radio-button>
            <paper-radio-button name="jp">Jurassic Park</paper-radio-button>
            <paper-radio-button name="td">Tenacious D in The Pick of Destiny</paper-
↪radio-button>
          </paper-radio-group>
        </paper-material>

      </div>
```

```
    </section>

    <paper-material elevation="1">
      <paper-slider name="slider 1"
                    value="{{slider1}}"></paper-slider>
      <paper-slider name="slider 2"
                    value="{{slider2}}"
                    max="100"
                    editable></paper-slider>
      <paper-slider name="slider 3"
                    pin
                    value="{{slider3}}"></paper-slider>
      <paper-slider name="slider 4"
                    max-markers="100"
                    pin
                    snaps
                    max="300"
                    step="20"
                    value="{{slider4}}"></paper-slider>
    </paper-material>

    <paper-fab icon="send" title="send" on-click="submit"></paper-fab>

    <h1>server response:</h1>

    <template is="dom-repeat" items="[[response]]" as="item">
      <p>[[item.name]] = [[item.value]]</p>
    </template>

  </template>
  <script src="javascript/form-example-min.js?__inline=true"></script>
</dom-module>
```

Now edit /src/html/elements/form-example/css/form-example.scss

/src/html/elements/form-example/css/form-example.scss

```
:host {
  display: block;
}

paper-material {
  margin: 1em;
  padding: 1em;
}

paper-fab {
  width: 5em;
  height: 5em;
  position: absolute;
  bottom: 1em;
  right: 1em;
}

p[pineapple-note] {
  border: 3px solid red;
  padding: 5px;
}
```

Now edit /src/html/elements/form-example/javascript/form-example.js

/src/html/elements/form-example/javascript/form-example.js

```
Polymer({
    is: 'form-example',

    properties: {
      response: {
        type: String,
        value: undefined
      },
      text2: {
        type: String,
        value: "default value"
      },
      withCheese: {
        type: Boolean,
        value: true
      },
      withSalami: {
        type: Boolean,
        value: false
      },
      withPineapple: {
        type: Boolean,
        value: false
      },
      withOnion: {
        type: Boolean,
        value: false
      },
      withKebab: {
        type: Boolean,
        value: true
      },
      slider2: {
        type: Number,
        value: 10
      },
      slider3: {
        type: Number,
        value: 20
      },
      slider4: {
        type: Number,
        value: 100
      },
      favoriteTVstuff: {
        type: String,
        value: "jp"
      },


      animationConfig: {
          type: Object,
          value: function() {
            return {
              'fadein': [{
```

```
                name: 'fade-in-animation',
                node: this,
                  timing: {duration: 500}
              }],
                'cascade':[ {
                  name: 'cascaded-animation',
                  animation: 'transform-animation',
                  transformFrom: 'translateY(100%)',
                  transformTo: 'none'
                }]
            }
          }
      }
    },

    submit: function() {
      this.$.ajax.generateRequest();
    },


    attached: function() {
      // creates the animation during panel-load
      this.async(function() {
        var nodeList = Polymer.dom(this.root).querySelectorAll('paper-material');
        this.animationConfig['cascade'][0].nodes = Array.prototype.slice.
↪call(nodeList);
        this.style.display = 'block';
        this.playAnimation('fadein');
        this.playAnimation('cascade');
      });
    },

    behaviors: [
      Polymer.NeonSharedElementAnimatableBehavior,
      Polymer.NeonAnimatableBehavior,
      Polymer.NeonAnimationRunnerBehavior
    ]
});
```

Now execute Grunt to build our new Polymer element.

```
cd src/html
grunt
```

Make the form-example panel

Make a new c++ file /src/common/panels/FormExample.cc

/src/common/panels/FormExample.cc

```
#include "subsystem/supervisor/panels/FormExample.h"
#include "ajax/PolymerElement.h"
#include "ajax/toolbox.h"
#include "json/json.h"

using namespace subsystempanels;
FormExample::FormExample( tsframework::CellAbstractContext* context, log4cplus::
↪Logger& logger)
```

```
:tsframework::CellPanel(context,logger) {
  logger_ = log4cplus::Logger::getInstance(logger.getName() +".FormExample");
}
FormExample::~FormExample() {
  remove();
}

void FormExample::layout(cgicc::Cgicc& cgi) {
  remove();
  setEvent("submit", ajax::Eventable::OnClick, this, &FormExample::submit);
  add(new ajax::PolymerElement("form-example"));
}

void FormExample::submit(cgicc::Cgicc& cgi,std::ostream& out) {
    std::map<std::string,std::string> values(ajax::toolbox::getSubmittedValues(cgi));

    Json::Value root(Json::arrayValue);

    for(std::map<std::string,std::string>::iterator i(values.begin()); i != values.
↪end(); ++i) {
      Json::Value someresult;
      someresult["name"] = i->first;
      someresult["value"] = i->second;
      root.append(someresult);
    }

    out << root;
}
```

Make the /include/subsystem/supervisor/panels/FormExample.h file.

/include/subsystem/supervisor/panels/FormExample.h

```
#ifndef _subsystem_supervisor_panels_FormExample_h_
#define _subsystem_supervisor_panels_FormExample_h_

#include "ts/framework/CellPanel.h"
#include "log4cplus/logger.h"
#include "cgicc/Cgicc.h"

namespace subsystempanels {
  class FormExample: public tsframework::CellPanel {
    public:
      FormExample(tsframework::CellAbstractContext* context, log4cplus::Logger&␣
↪logger);
                          ~FormExample();
      void layout(cgicc::Cgicc& cgi);
                private:
                 void submit(cgicc::Cgicc& cgi,std::ostream& out);
  };
}
#endif
```

Register the new class in the Makefile.

```
Sources=\
    version.cc\
    Cell.cc\
```

```
    CellContext.cc\
    Configuration.cc\
    ...
    panels/FormExample.cc \
    ...
```

Now register your new panel in the menu so users can access it.

Edit src/common/Cell.cc

```
#include "subsystem/supervisor/panels/FormExample.h"
...
void subsystemsupervisor::Cell::init() {
  ...
  tsframework::CellPanelFactory* panelF = getContext()->getPanelFactory();
  ...
  panelF->add<subsystempanels::FormExample>("Form example");
```

Now you can compile your cell and you should see the "Form example" panel in the menu under the 'control-panels' section.

Also your element has created some documentation. Surf to <hostname>:<port>/<package-name>/html/index.html and you will see the package documentation for your cell. *form-example* will be in there, and clicking it brings up the documentation for your *form-example* element.

1.2.7 Demo 4: Refresh

The *refresh-example* element behaves much like the *ts-ajax* element we saw in the previous demo. The big difference is that this element implements periodic updating.

Make the refresh-example element

In your cell, run:

```
cd src/html/elements
./new-element.js
name of the new element: refresh-example
creating new element <refresh-example>...
removing any .svn folders in  refresh-example
Finished
```

Register the refresh-example element

Edit src/html/elements/elements.html and add the following line

```
<link rel="import" href="refresh-example/refresh-example.html">
```

This will tell AjaXell to load our new element.

Edit the refresh-example element

Edit src/html/elements/refresh-example/refresh-example.html

/src/html/elements/refresh-example/refresh-example.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/extern/bower_components/iron-media-query/iron-media-query.
↪html">
<link rel="import" href="/extern/bower_components/paper-checkbox/paper-checkbox.html">
<link rel="import" href="/extern/bower_components/paper-input/paper-input.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/ts/common-elements/auto-update/auto-update.html">
<link rel="import" href="/ts/common-elements/iron-flex-layout-attributes/iron-flex-
↪layout-attributes.html">


<dom-module id="refresh-example">
  <template>
    <link rel="import" type="css" href="css/refresh-example-min.css?__inline=true">
    <style include="iron-flex-layout-attributes"></style>
    <style include="reset-css"></style>

    <paper-checkbox checked="{{refreshtoggle}}">Pauze refresh</paper-checkbox>
    <br>
    <auto-update data="{{test}}"
                 interval="1000"
                 callback="RefreshExample::timesExecuted"
                 handle-as="text"
                 timeout="10000"
                 no-refresh$={{refreshtoggle}}></auto-update>
    <span>{{test}}</span>

    <hr>

    <paper-input value="{{favDino}}" label="your favorite dinosaur"></paper-input>
    <auto-update data="{{test2}}"
                 interval="1000"
                 callback="RefreshExample::timesExecutedWithParameters"
                 handle-as="text"
                 parameters='["favoritedino"]'
                 favoritedino="{{favDino}}"></auto-update>
    <!-- notice the single & double quotes in the parameters variable
    do yourself a favor and do not use capital letters in parameter names -->
    <span>{{test2}}</span>
  </template>
  <script src="javascript/refresh-example-min.js?__inline=true"></script>
</dom-module>
```

Now edit /src/html/elements/refresh-example/css/refresh-example.scss

/src/html/elements/refresh-example/css/refresh-example.scss

```
:host {
    color: black;
}
```

Now edit /src/html/elements/refresh-example/javascript/refresh-example.js

/src/html/elements/refresh-example/javascript/refresh-example.js

```
Polymer({
    is: "refresh-example",
    properties: {
      test: String,
```

```
      favDino: {
        type: String,
        value: "Velociraptor"
      }
    },
    doTimesExecuted: function() {
      var parameters = {};
      parameters._eventType_ = "OnClick";
      parameters._id_ = "RefreshExample-timesExecuted";
      this.$.ajax_timesExecuted.params = parameters;
      this.$.ajax_timesExecuted.url = window.location.origin + window.location.
↪pathname;
      this.$.ajax_timesExecuted.generateRequest();
    },
    timesExecutedResponse: function(event, detail, sender) {
      console.log(detail.response);
      Polymer.dom(this.$.timesExecutedContainer).innerHTML = detail.response;
    }
});
```

Now execute Grunt to build our new Polymer element.

```
cd src/html
grunt
```

Make the refresh-example panel

Make a new c++ file /src/common/panels/RefreshExample.cc

/src/common/panels/RefreshExample.cc

```
/*****************************************************************************
 * Trigger Supervisor Component                                              *
 *                                                                           *
 * Project Manager: Ildfons Magrans de Abril                                   *
 *                                                                           *
 * developer: Marc Magrans de Abril                                            *
 *****************************************************************************/
#include "subsystem/supervisor/panels/RefreshExample.h"

#include "ajax/PolymerElement.h"
#include "ajax/toolbox.h"

#include <iostream>
using namespace subsystempanels;
RefreshExample::RefreshExample(tsframework::CellAbstractContext* context, log4cplus::
↪Logger& logger)
        : tsframework::CellPanel(context,logger)
{
  logger_ = log4cplus::Logger::getInstance(logger.getName() +".RefreshExample");

}

RefreshExample::~RefreshExample()
{
  ;
}
```

```

void RefreshExample::timesExecuted(cgicc::Cgicc& cgi,std::ostream& out)
{
  static int times(0);
  times++;
        std::ostringstream msg;
        msg << "The RefreshExample::timesExecuted method has been executed " << times
↪<< " times";
        out << msg.str();
}
void RefreshExample::timesExecutedWithParameters(cgicc::Cgicc& cgi,std::ostream& out)
{
        std::ostringstream msg;

        std::map<std::string,std::string> values(ajax::toolbox::
↪getSubmittedValues(cgi));

        msg << "Your favorite dinosaur is the ";

        for(std::map<std::string,std::string>::iterator i(values.begin()); i !=␣
↪values.end(); ++i) {
                        if ( i->first.compare("favoritedino") == 0 ) {
                                msg << i->second;
                        }
        }

        out << msg.str();
}


void RefreshExample::layout(cgicc::Cgicc& cgi)
{
        //clear the output buffer of this panel, we start fresh
  remove();

        setEvent("RefreshExample::timesExecuted", ajax::Eventable::OnTime, this, &
↪RefreshExample::timesExecuted);
        setEvent("RefreshExample::timesExecutedWithParameters", ajax::Eventable::
↪OnTime, this, &RefreshExample::timesExecutedWithParameters);

        // ajax::PolymerElement* element = ;
        add(new ajax::PolymerElement("refresh-example"));

}
```

Make the /include/subsystem/supervisor/panels/RefreshExample.h file.

/include/subsystem/supervisor/panels/RefreshExample.h

```
/*************************************************************************
 * Trigger Supervisor Component                                          *
 *                                                                       *
 * Project Manager: Ildfons Magrans de Abril                             ␣
↪      *
 * Authors: Marc Magrans de Abril                                        ␣
↪    *
 *************************************************************************/

```

```
#ifndef _subsystem_supervisor_panels_RefreshExample_h_
#define _subsystem_supervisor_panels_RefreshExample_h_

#include "ts/framework/CellPanel.h"

#include "ts/framework/CellAbstractContext.h"

#include "log4cplus/logger.h"

#include "cgicc/Cgicc.h"

#include <iostream>

namespace subsystempanels
{

class RefreshExample: public tsframework::CellPanel
{
 public:
  RefreshExample(tsframework::CellAbstractContext* context, log4cplus::Logger&
↪logger);
  ~RefreshExample();

  void layout(cgicc::Cgicc& cgi);

 private:
  void timesExecuted(cgicc::Cgicc& cgi,std::ostream& out);
  void timesExecutedWithParameters(cgicc::Cgicc& cgi,std::ostream& out);
};
}//ns subsystempanels
#endif
```

Register the new class in the Makefile.

```
Sources=\
    version.cc\
    Cell.cc\
    CellContext.cc\
    Configuration.cc\
    ...
    panels/RefreshExample.cc \
    ...
```

Now register your new panel in the menu so users can access it.

Edit src/common/Cell.cc

```
#include "subsystem/supervisor/panels/RefreshExample.h"
...
void subsystemsupervisor::Cell::init() {
  ...
  tsframework::CellPanelFactory* panelF = getContext()->getPanelFactory();
  ...
  panelF->add<subsystempanels::RefreshExample>("Refresh example");
```

Now you can compile your cell and you should see the “Form example” panel in the menu under the ‘control-panels’ section.

Also your element has created some documentation. Surf to <hostname>:<port>/<package-name>/html/index.html

and you will see the package documentation for your cell. *refresh-example* will be in there, and clicking it brings up the documentation for your *refresh-example* element.

1.2.8 Demo 5: Tables

Make the table-example element

In your cell, run:

```
cd src/html/elements
./new-element.js
name of the new element: table-example
creating new element <table-example>...
removing any .svn folders in  table-example
Finished
```

Register the table-example element

Edit src/html/elements/elements.html and add the following line

```
<link rel="import" href="table-example/table-example.html">
```

This will tell AjaXell to load our new element.

Edit the table-example element

Edit src/html/elements/table-example/table-example.html

/src/html/elements/table-example/table-example.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/ts/common-elements/iron-flex-layout-attributes/iron-flex-
↪layout-attributes.html">
<link rel="import" href="/extern/bower_components/vaadin-grid/vaadin-grid.html">
<link rel="import" href="/extern/bower_components/paper-material/paper-material.html">

<!--
This panel uses [vaadin-grid](/extern/bower_components/vaadin-grid/index.html)

`table-example` demonstrates the uses of the vaadin-grid element.

-->
<dom-module id="table-example">
  <template>
    <style include="reset-css"></style>
    <style include="iron-flex-layout-attributes"></style>

    <link rel="stylesheet" type="text/css" href="css/table-example-min.css?__
↪inline=true">

  </template>
  <script src="javascript/table-example-min.js?__inline=true"></script>
</dom-module>
```

We'll add more stuff as we go along.

Now edit /src/html/elements/table-example/css/table-example.scss

/src/html/elements/table-example/css/table-example.scss

```
:host {
  display: block;
}

paper-material {
  margin: 1em;
}
```

Now execute Grunt to build our new Polymer element.

```
cd src/html
grunt
```

Make the table-example panel

Make a new c++ file /src/common/panels/TableExample.cc

/src/common/panels/TableExample.cc

```
#include "subsystem/supervisor/panels/TableExample.h"

#include "ajax/toolbox.h"
#include "ajax/PolymerElement.h"
#include "ajax/toolbox.h"
#include "json/json.h"

#include "log4cplus/loggingmacros.h"

#include <iostream>
#include <time.h>
#include <string>
using namespace subsystempanels;

TableExample::TableExample( tsframework::CellAbstractContext* context, log4cplus::
↪Logger& logger ) : tsframework::CellPanel(context,logger) {
    logger_ = log4cplus::Logger::getInstance(logger.getName() +".TableExample");
}
TableExample::~TableExample() {
    remove();
}

void TableExample::layout(cgicc::Cgicc& cgi)
{
    remove();
    add(new ajax::PolymerElement("table-example"));
}
```

Make the /include/subsystem/supervisor/panels/TableExample.h file.

/include/subsystem/supervisor/panels/TableExample.h

```
#ifndef _subsystem_supervisor_panels_TableExample_h_
#define _subsystem_supervisor_panels_TableExample_h_

#include "ts/framework/CellPanel.h"

#include "ts/framework/CellAbstractContext.h"

#include "log4cplus/logger.h"

#include "cgicc/Cgicc.h"

#include <iostream>

namespace subsystempanels
{

class TableExample: public tsframework::CellPanel
{
 public:
  TableExample(tsframework::CellAbstractContext* context, log4cplus::Logger& logger);
  ~TableExample();

  void layout(cgicc::Cgicc& cgi);

 private:

};
}//subsystempanels
#endif
```

Register the new class in the Makefile.

```
Sources=\
    version.cc\
    Cell.cc\
    CellContext.cc\
    Configuration.cc\
    ...
    panels/TableExample.cc \
    ...
```

Now register your new panel in the menu so users can access it.

Edit src/common/Cell.cc

```
#include "subsystem/supervisor/panels/TableExample.h"
...
void subsystemsupervisor::Cell::init() {
  ...
  tsframework::CellPanelFactory* panelF = getContext()->getPanelFactory();
  ...
  panelF->add<subsystempanels::TableExample>("Table example");
```

Now you can compile your cell and you should see the “Form example” panel in the menu under the ‘control-panels’ section.

Table1: A basic table with sorting

The first example will fetch a JSON file containing 1000 rows. Also we will make it sortable, so we will write our own sorting function.

C++ code

Edit /src/common/panels/TableExample.cc

/src/common/panels/TableExample.cc

```
void TableExample::layout(cgicc::Cgicc& cgi)
{
    remove();
    setEvent("getTable1", ajax::Eventable::OnClick, this, &TableExample::getTable1);
    add(new ajax::PolymerElement("table-example"));
}

void TableExample::getTable1(cgicc::Cgicc& cgi,std::ostream& out) {
    Json::Value root;

    Json::Value items(Json::arrayValue);
    for (int i = 0; i < 1000; i++) {
      Json::Value row;

      std::ostringstream msg;
      msg << "row" << i;
      row["some string"] = msg.str();

      msg << "@cern.ch";
      row["email"] = msg.str();

      row["group"] = i % 3;

      items.append(row);
    }
    root["items"] = items;

    Json::Value columns(Json::arrayValue);

    Json::Value column1;
    column1["name"] = "some string";
    column1["sortable"] = true;
    columns.append(column1);

    Json::Value column2;
    column2["name"] = "email";
    column2["sortable"] = true;
    columns.append(column2);

    Json::Value column3;
    column3["name"] = "group";
    column3["sortable"] = true;
    columns.append(column3);

    root["columns"] = columns;

```

```
    out << root;
}
```

The getTable1 call will provide the data of our first table. It will provide both column definitions and table rows.

Edit /include/subsystem/supervisor/panels/TableExample.h.

/include/subsystem/supervisor/panels/TableExample.h

```
namespace subsystempanels
{

class TableExample: public tsframework::CellPanel
{

 private:
  void getTable1(cgicc::Cgicc& cgi,std::ostream& out);

};
}//subsystempanels
#endif
```

HTML

Edit src/html/elements/table-example/table-example.html

/src/html/elements/table-example/table-example.html

```
<dom-module id="table-example">
  <template>

    <h1>basic table, also sortable</h1>
    <p>
      This dataset has 1000 records, however this table is smart and will only render␣
↪the rows currently in view.
      This will keep the page performant.
    </p>
    <p>
      Click on one of the colums to sort it, shift + click a column to use secondary␣
↪sort
    </p>
    <paper-material elevation="1">
      <ts-ajax data="{{table1}}" callback="getTable1" handle-as="json" auto></ts-ajax>
      <vaadin-grid id="table1"
                   selection-mode="multi"
                   items="{{table1.items}}"
                   columns="{{table1.columns}}"></vaadin-grid>
    </paper-material>

  </template>
</dom-module>
```

ts-ajax will execute the getTable1 callback (notice the 'auto' attribute). the table will populate itself as soon as ts-ajax completed the callback.

Making it sortable

In the C++ code, our generated JSON contains the "sortable": true, but we still have to supply a sorting function ourselves.

This gets a bit complicated as we made more than one column sortable, so we have to support secondary sorting (the user can shift + click on a column to do that), and we have data that we need to parse to make properly sortable (like the email column).

Edit /src/html/elements/table-example/javascript/table-example.js

/src/html/elements/table-example/javascript/table-example.js

```
Polymer({
  attached: function() {
    this.$.table1.addEventListener('sort-order-changed', this.table1sort.bind(this));
  },

  table1sort: function() {
    // secondary sort
    var secondarySort = function() {return 0};
    if (this.$.table1.sortOrder[1]) {
      var columnNameSecondary = this.$.table1.columns[this.$.table1.sortOrder[1].
↪column].name;
      var directionSecondary = this.$.table1.sortOrder[1].direction == 'asc' ? -1 : 1;
      secondarySort = function(row1, row2) {
        return (row1[columnNameSecondary] < row2[columnNameSecondary]) ?␣
↪directionSecondary : -directionSecondary;
      }

      if (columnNameSecondary == "some string") {
        secondarySort = function(row1, row2) {
          var a = parseInt(row1[columnNameSecondary].substring(3));
          var b = parseInt(row2[columnNameSecondary].substring(3));
          return (a < b) ? directionSecondary : -directionSecondary;
        }
      }
      if (columnNameSecondary == "email") {
        secondarySort = function(row1, row2) {
          var a = parseInt( row1[columnNameSecondary].substring(3).substring(0,␣
↪row1[columnNameSecondary].indexOf("@") - 3) );
          var b = parseInt( row2[columnNameSecondary].substring(3).substring(0,␣
↪row2[columnNameSecondary].indexOf("@") - 3) );
          return (a < b) ? directionSecondary : -directionSecondary;
        }
      }
    }

    // primary sort
    var columnName = this.$.table1.columns[this.$.table1.sortOrder[0].column].name;
    var direction = this.$.table1.sortOrder[0].direction == 'asc' ? -1 : 1;

    var sort = function(row1, row2) {
      var result = (row1[columnName] < row2[columnName]) ? direction : -direction;
      if (row1[columnName] == row2[columnName]) {
        result = secondarySort(row1, row2);
      }
      return result;
    }
```

```
43
44      if (columnName == "some string") {
45        sort = function(row1, row2) {
46          var a = parseInt(row1[columnName].substring(3));
47          var b = parseInt(row2[columnName].substring(3));
48          var result = (a < b) ? direction : -direction;
49          if (a == b) {
50            result = secondarySort(row1, row2);
51          }
52          return result;
53        }
54      }
55      if (columnName == "email") {
56        sort = function(row1, row2) {
57          var a = parseInt( row1[columnName].substring(3).substring(0, row1[columnName].
    ↪indexOf("@") - 3) );
58          var b = parseInt( row2[columnName].substring(3).substring(0, row2[columnName].
    ↪indexOf("@") - 3) );
59          var result = (a < b) ? direction : -direction;
60          if (a == b) {
61            result = secondarySort(row1, row2);
62          }
63          return result;
64        }
65      }
66
67      this.$.table1.items.sort(sort);
68    }
69  });
```

The result will look like this:

basic table, also sortable

This dataset has 1000 records, however this table is smart and will only render the rows currently in view. This will keep the page performant.

Click on one of the colums to sort it, shift + click a column to use secondary sort

	Some String ↓₂	Email	Group ↑₁
☐	row15	row15@cern.ch	0
☐	row12	row12@cern.ch	0
☐	row9	row9@cern.ch	0
☐	row6	row6@cern.ch	0
☐	row3	row3@cern.ch	0
☐	row0	row0@cern.ch	0
☐	row997	row997@cern.ch	1
☐	row994	row994@cern.ch	1
☐	row991	row991@cern.ch	1
☐	row988	row988@cern.ch	1

Table2: An AJAX table

This dataset will be 5000 rows big, and is loaded asynchronously. This means the table will make requests for more data as the user scrolls.

C++ code

Edit /src/common/panels/TableExample.cc

/src/common/panels/TableExample.cc

```
void TableExample::layout(cgicc::Cgicc& cgi)
{
    remove();
    setEvent("getTable1", ajax::Eventable::OnClick, this, &TableExample::getTable1);
    setEvent("getTable2", ajax::Eventable::OnClick, this, &TableExample::getTable2);
    add(new ajax::PolymerElement("table-example"));
}

void TableExample::getTable2(cgicc::Cgicc& cgi,std::ostream& out) {
  std::map<std::string,std::string> values(ajax::toolbox::getSubmittedValues(cgi));
  int index;
  std::stringstream indexs(values.find("index")->second);
  indexs >> index;
  int count;
  std::stringstream counts(values.find("count")->second);
  counts >> count;
  if (indexs.fail() || counts.fail()) {
      return;
  }

  Json::Value rows(Json::arrayValue);
  for (size_t i = index; i <= count + index; i++) {
    Json::Value row;

    row["random number"] = rand();
    std::ostringstream msg;
    msg << "item " << i;
    row["some string"] = msg.str();

    rows.append(row);
  }

  out << rows;
}
```

Edit /include/subsystem/supervisor/panels/TableExample.h.

/include/subsystem/supervisor/panels/TableExample.h

```
namespace subsystempanels
{

class TableExample: public tsframework::CellPanel
{

 private:
  void getTable2(cgicc::Cgicc& cgi,std::ostream& out);
```

```
};
}//subsystempanels
#endif
```

Unlike the first example, were we sent both items and column info, we only send the items in this example. The column info will now be declared in JavaScript.

HTML & Javascript

Edit src/html/elements/table-example/table-example.html

/src/html/elements/table-example/table-example.html

```
<dom-module id="table-example">
  <template>

    <h1>Asynchronous table</h1>
    <p>
      This dataset is not fetched in one go from the server, but in chunks.
      Use this if the data is expensive to render server side.
      Scroll fast to see the data being loaded.
    </p>
    <p>
      Unfortunately, async data means we cannot use sorting...
    </p>
    <paper-button raised primary on-click="table2_scrollend">Scroll to end</paper-
↪button>
    <paper-button raised primary on-click="table2_scrollstart">Scroll to start</paper-
↪button>
    <paper-button raised primary on-click="table2_scroll3000">Scroll to line 3000</
↪paper-button>
    <paper-material elevation="1">
      <ts-ajax id="ajax_table2"
               data="{{tsajax_table2}}"
               callback="getTable2"
               handle-as="json"
               parameters='["index", "count"]'></ts-ajax>
      <vaadin-grid id="table2"
                   selection-mode="multi"
                   items="{{table2items}}"
                   columns='{{table2columns}}'
                   size="5000"></vaadin-grid>
    </paper-material>

  </template>
</dom-module>
```

Now edit /src/html/elements/table-example/javascript/table-example.js

/src/html/elements/table-example/javascript/table-example.js

```
Polymer({
  properties: {
    /**
     * The callback for the asynchronous data request
     */
```

```
    table2callback: Function,
    /**
     * The ajax response from ts-ajax
     */
    tsajax_table2: {
      type: Object,
      observer: 'newTable2data'
    },
    /**
     * The `items` dataset for our table. It is a function because it will fetch
     * data for us rather than just be a dumb array containing all the data.
     */
    table2items: {
      type: Function,
      value: function() {
        return function(params, callback) {
          this.table2callback = callback;
          var ajax = this.$.ajax_table2;
          ajax.index = params.index;
          ajax.count = params.count;
          ajax.generateRequest();
        }.bind(this);
      }
    },
    /**
     * The columns of our data
     */
    table2columns: {
      type: Array,
      value: function() {
        return [{name: "some string"}, {name: "random number"}]
      }
    }
  },

  newTable2data: function(newdata) {
    if (this.table2callback) {
      // note that the callback can also take a second parameter that updates the size
      // this can be used to implement infinite scrolling or datasets with changing sizes
      this.table2callback(newdata);
    }
  },

  table2_scrollend: function() {
    this.$.table2.scrollToEnd();
  },
  table2_scrollstart: function() {
    this.$.table2.scrollToStart();
  },
  table2_scroll3000: function() {
    this.$.table2.scrollToRow(3000);
  },
});
```

Notice the asynchronous data fetching makes our JavaScript quite a bit more complex. Use this approach when generating data on server-side is slow or otherwise expensive.

The result will look like this:

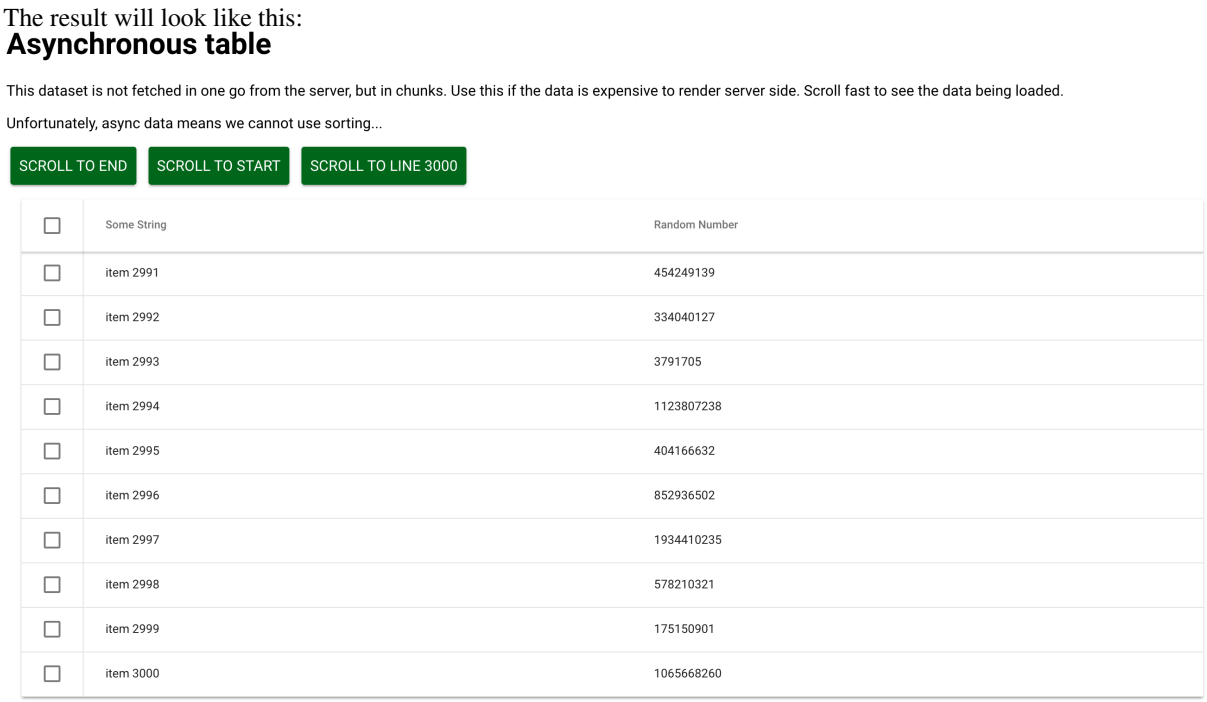


Table3: Frozen columns & styling

This table will have a 'frozen' column that will always be visible. Also some columns are 'hidable', so we will have the option to hide them.

C++ code

Edit /src/common/panels/TableExample.cc

/src/common/panels/TableExample.cc

```
void TableExample::layout(cgicc::Cgicc& cgi)
{
    remove();
    setEvent("getTable1", ajax::Eventable::OnClick, this, &TableExample::getTable1);
    setEvent("getTable2", ajax::Eventable::OnClick, this, &TableExample::getTable2);
    setEvent("getTable3", ajax::Eventable::OnClick, this, &TableExample::getTable3);
    add(new ajax::PolymerElement("table-example"));
}

void TableExample::getTable3(cgicc::Cgicc& cgi,std::ostream& out) {
  std::map<std::string,std::string> values(ajax::toolbox::getSubmittedValues(cgi));
  int index;
  std::stringstream indexs(values.find("index")->second);
  indexs >> index;
  int count;
  std::stringstream counts(values.find("count")->second);
  counts >> count;
  if (indexs.fail() || counts.fail()) {
      return;
```

```
  }

  std::string states[] = {"ok", "warning", "error"};
  Json::Value rows(Json::arrayValue);
  for (size_t i = index; i <= count + index; i++) {
    Json::Value row;

    std::ostringstream msg;
    msg << "cell #" << i;
    row["cell"] = msg.str();

    row["CPU (%)"] = rand() % 100;
    row["memory (%)"] = rand() % 100;
    row["network (%)"] = rand() % 100;
    row["PID"] = rand() % 700;
    row["RSS"] = rand();
    row["VSZ"] = rand();
    row["state"] = states[i % 3];

    time_t      now = time(0);
    struct tm  tstruct;
    char       buf[80];
    tstruct = *localtime(&now);
    // Visit http://en.cppreference.com/w/cpp/chrono/c/strftime
    // for more information about date/time format
    strftime(buf, sizeof(buf), "%Y-%m-%d.%X", &tstruct);
    row["timestamp"] = buf;

    rows.append(row);
  }

  out << rows;
}
```

Edit /include/subsystem/supervisor/panels/TableExample.h.

/include/subsystem/supervisor/panels/TableExample.h

```
namespace subsystempanels
{

class TableExample: public tsframework::CellPanel
{

 private:
  void getTable3(cgicc::Cgicc& cgi,std::ostream& out);

};
}//subsystempanels
#endif
```

HTML & Javascript

Edit src/html/elements/table-example/table-example.html

/src/html/elements/table-example/table-example.html

```
<dom-module id="table-example">
  <template>

    <h1>Frozen columns &amp; styling</h1>
    <p>
      The first column in this dataset will always be visible.
      Errors and warnings will be very visible.
    </p>
    <p>
      You can also choose to hide some columns using the icon in the upper right.
    </p>
    <paper-material elevation="1">
      <ts-ajax id="ajax_table3"
             data="{{tsajax_table3}}"
             callback="getTable3"
             handle-as="json"
             parameters='["index", "count"]'></ts-ajax>
      <vaadin-grid id="table3"
                   selection-mode="multi"
                   items="{{table3items}}"
                   columns='{{table3columns}}'
                   size="5000"
                   row-class-generator="[[table3rowclass]]"
                   frozen-columns="1"></vaadin-grid>
    </paper-material>

  </template>
</dom-module>
```

Now edit /src/html/elements/table-example/javascript/table-example.js

/src/html/elements/table-example/javascript/table-example.js

```
                          .       .
                         / `.   .' \
                 .---.  <    > <    >  .---.
                 |    \  \ - ~ ~ - /  /    |
                  ~-..-~             ~-..-~
              \~~~\.'                    `./~~~/
               \__/                        \__/
                /                  .-    .  \
         _._ _.-    .-~ ~-.       /       \}   \/~~~/
    _.-'q  \}~     /       \}     \{        ;    \__/
   \{'__,  /      (       /      \{       /      `. ,~~|   .     .
    `''''='~~-.__(      /_      |      /- _      `..-'   \\   //
                / \   =/  ~~--~~\{    ./|    ~-.     `-..__\\_//_.-'
               \{   \  +\         \  =\ (        ~ - . _ _ _..---~
               |  | \{   \}         \   \_\
              '---.o___,'       .o___,'
Polymer({
  properties: {
    /**
     * The callback for the asynchronous data request
     */
    table3callback: Function,
    /**
     * The ajax response from ts-ajax
     */
```

```
    tsajax_table3: {
      type: Object,
      observer: 'newTable3data'
    },
    /**
     * The `items` dataset for our table. It is a function because it will fetch
     * data for us rather than just be a dumb array containing all the data.
     */
    table3items: {
      type: Function,
      value: function() {
        return function(params, callback) {
          this.table3callback = callback;
          var ajax = this.$.ajax_table3;
          ajax.index = params.index;
          ajax.count = params.count;
          ajax.generateRequest();
        }.bind(this);
      }
    },
    /**
     * The columns of our data
     */
    table3columns: {
      type: Array,
      value: function() {
        return [
          {name: "cell"},
          {name: "CPU (%)", hidable: true},
          {name: "memory (%)", hidable: true},
          {name: "network (%)", hidable: true},
          {name: "PID", hidable: true},
          {name: "RSS", hidable: true},
          {name: "VSZ", hidable: true},
          {name: "state"},
          {name: "timestamp", hidable: true}
        ];
      }
    },

    table3rowclass: {
      type: Function,
      value: function() {
        return function(row) {
          var bgcolor = "";
          var color = "";
          // var fontweight = "";
          var DOMrow = row.element;

          if (row.data.state == "error") {
            bgcolor = "#EC4343";
            color = "white";
            DOMrow.childNodes[8].style.textTransform = "uppercase";
          } else if (row.data.state == "warning") {
            bgcolor = "#EC9943";
            color = "white";
            DOMrow.childNodes[8].style.textTransform = "uppercase";
          }
```

```
84
85            DOMrow.style.color = color;
86            for (var i = 0; i < DOMrow.childNodes.length; i++) {
87              DOMrow.childNodes[i].style.backgroundColor = bgcolor;
88            }
89
90            return "";
91          }
92        }
93      }
94    },
95
96    newTable3data: function(newdata) {
97      if (this.table3callback && newdata) {
98        this.table3callback(newdata);
99      }
100   }
101
102 });
```

The result will look like this:

Frozen columns & styling

The first 2 columns in this dataset will always be visible. Errors and warnings will be very visible.

You can also choose to hide some columns using the icon in the upper right.

Cell	CP U (%)	Memory (%)	Network (%)	PID	RSS	VSZ	State	Timestamp
cell #0	56	98	11	69	1143739824	601430824	OK	2016-04-07.14:52:10
cell #1	93	74	88	526	175664460	100668754	WARNING	2016-04-07.14:52:10
cell #2	58	84	77	544	255169849	951214646	ERROR	2016-04-07.14:52:10
cell #3	33	25	44	194	698797658	874485379	OK	2016-04-07.14:52:10
cell #4	60	40	35	458	798758151	24551057	WARNING	2016-04-07.14:52:10
cell #5	83	62	99	360	41542630	857441135	ERROR	2016-04-07.14:52:11
cell #6	16	71	48	492	1644694092	1513223639	OK	2016-04-07.14:52:11
cell #7	91	69	0	633	183946714	1117362635	WARNING	2016-04-07.14:52:11
cell #8	82	60	72	629	2025758855	696556657	ERROR	2016-04-07.14:52:11
cell #9	12	28	36	473	1518401987	1368301838	OK	2016-04-07.14:52:11

Table4: Custom cell content

Sometimes you want to display something fancy instead of just data. This example will replace the 'progress' value (0-100) with an html progress bar.

C++ code

Edit /src/common/panels/TableExample.cc

/src/common/panels/TableExample.cc

```
void TableExample::layout(cgicc::Cgicc& cgi)
{
    remove();
    setEvent("getTable1", ajax::Eventable::OnClick, this, &TableExample::getTable1);
    setEvent("getTable2", ajax::Eventable::OnClick, this, &TableExample::getTable2);
    setEvent("getTable3", ajax::Eventable::OnClick, this, &TableExample::getTable3);
    setEvent("getTable4", ajax::Eventable::OnClick, this, &TableExample::getTable4);
    add(new ajax::PolymerElement("table-example"));
}

void TableExample::getTable4(cgicc::Cgicc& cgi,std::ostream& out) {
    Json::Value root;

    Json::Value items(Json::arrayValue);
    for (int i = 1; i <= 100; i++) {
      Json::Value row;

      std::ostringstream msg;
      msg << "row " << i;
      row["rowNumber"] = msg.str();

      row["progress"] = rand() % 100;

      row["extra info"] = "some extra info about row #" + msg.str();

      items.append(row);
    }
    root["items"] = items;

    Json::Value columns(Json::arrayValue);

    Json::Value column1;
    column1["name"] = "rowNumber";
    columns.append(column1);

    Json::Value column3;
    column3["name"] = "progress";
    columns.append(column3);

    root["columns"] = columns;

    out << root;
}
```

Edit /include/subsystem/supervisor/panels/TableExample.h.

/include/subsystem/supervisor/panels/TableExample.h

```
namespace subsystempanels
{

class TableExample: public tsframework::CellPanel
{

 private:
  void getTable4(cgicc::Cgicc& cgi,std::ostream& out);

};
```

```
}//subsystempanels
#endif
```

HTML & Javascript

Edit src/html/elements/table-example/table-example.html

/src/html/elements/table-example/table-example.html

```
<dom-module id="table-example">
  <template>

    <h1>Custom HTML instead of pure data</h1>
    <p>
      Instead of some progress number (e.g. 10 or 80), you can show a progress bar.
    </p>
    <paper-material elevation="1">
      <ts-ajax data="{{table4}}" callback="getTable4" handle-as="json" auto></ts-ajax>
      <vaadin-grid id="table4"
                   items="{{table4.items}}"
                   columns="{{table4.columns}}"></vaadin-grid>
    </paper-material>

  </template>
</dom-module>
```

Now edit /src/html/elements/table-example/javascript/table-example.js

/src/html/elements/table-example/javascript/table-example.js

```
Polymer({
  properties: {
    table4: {
      type: Object,
      observer: "table4Changed"
    }
  },

  table4Changed: function(table4) {
    if (table4) {
      table4.columns[1].renderer = this.table4ProgressRenderer;
    }
  },

  table4ProgressRenderer: function(cell) {
    cell.element.innerHTML = '';
    var child = document.createElement('progress');
    child.setAttribute('value', cell.data);
    child.setAttribute('max', 100);
    cell.element.appendChild(child);
  }

});
```

The result will look like this:

Custom HTML instead of pure data

Instead of some progress number (e.g. 10 or 80), you can show a progress bar.

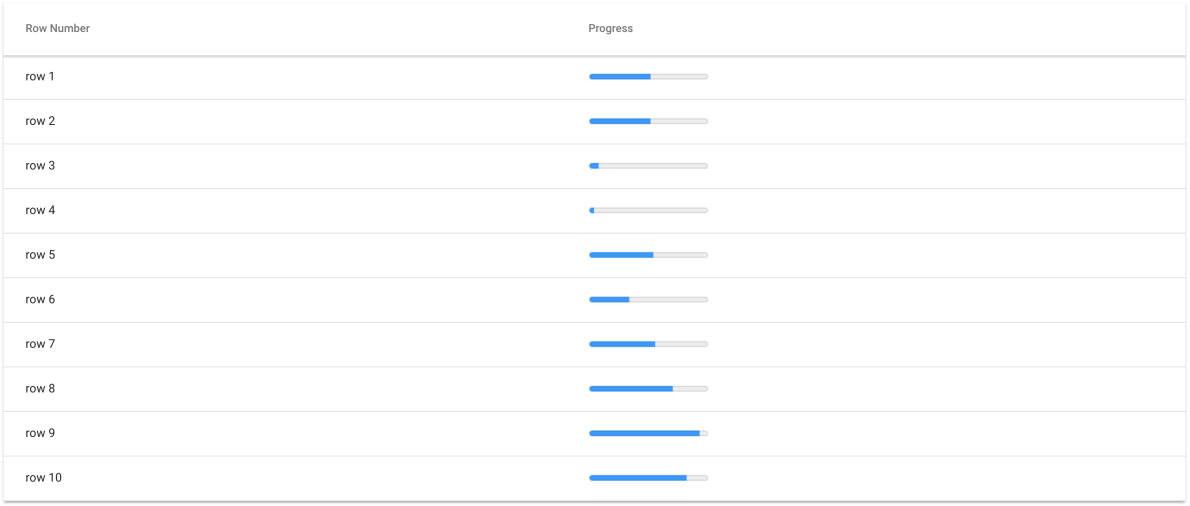


Table5: Details on selection

You can show some extra details when one of the rows is clicked.

HTML & Javascript

Edit src/html/elements/table-example/table-example.html

/src/html/elements/table-example/table-example.html

```
<dom-module id="table-example">
  <template>

    <h1>details on selection</h1>
    <p>
      Select an item, and a detail view will appear
    </p>
    <paper-material elevation="1">
      <ts-ajax data="{{table5}}" callback="getTable4" handle-as="json" auto></ts-ajax>
      <vaadin-grid id="table5"
                   items="{{table5.items}}"
                   columns="{{table5.columns}}"></vaadin-grid>
    </paper-material>

  </template>
</dom-module>
```

Now edit /src/html/elements/table-example/javascript/table-example.js

/src/html/elements/table-example/javascript/table-example.js

```
Polymer({
  properties: {
    table5_detailSelected: {
      type: Number,
```

```
      value: 0
    }
  },

  attached: function() {

    this.$.table5.addEventListener('selected-items-changed', function() {
      this.$.table5.setRowDetailsVisible(this.table5_detailSelected, false);
      var selected = this.$.table5.selection.selected();
      if (selected.length == 1) {
        this.$.table5.setRowDetailsVisible(selected[0], true);
        this.table5_detailSelected = selected[0];
      }
    }.bind(this));

    this.$.table5.rowDetailsGenerator = function(rowIndex) {
      var elem = document.createElement('table-detail');

      this.$.table5.getItem(rowIndex, function(error, item) {
        if (!error) {
          elem.item = item;
        }
      });

      return elem;
    }.bind(this);
  }

});
```

Now our code will show a 'table-detail' element when a row is selected. We still need to make this element.

Create src/html/elements/table-example/table-detail.html

/src/html/elements/table-example/table-detail.html

```
<link rel="import" href="/extern/bower_components/paper-material/paper-material.html">

<dom-module id="table-detail">
  <template>
    <link rel="stylesheet" type="text/css" href="css/table-detail-min.css?__
↪inline=true">

    <paper-material elevation="1">
      <p>
        [[getExtraInfo(item)]]
      </p>
    </paper-material>

  </template>
  <script src="javascript/table-detail-min.js?__inline=true"></script>
</dom-module>
```

Now create /src/html/elements/table-example/javascript/table-detail.js

/src/html/elements/table-example/javascript/table-detail.js

```
Polymer({
  is: "table-detail",
```

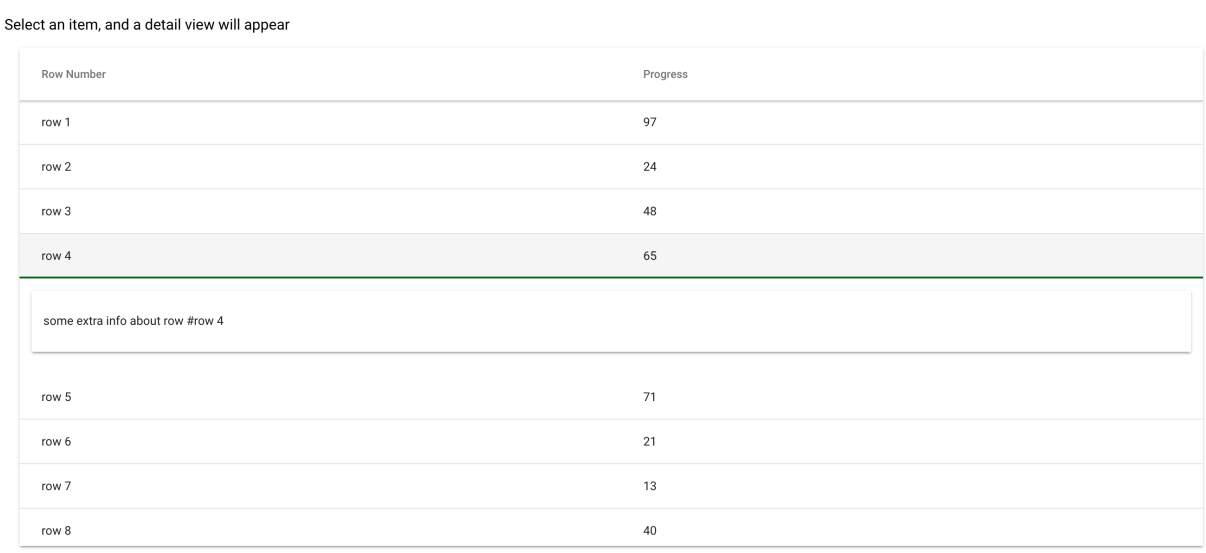
```
  properties: {
    item: {
      type: Object
    }
  },

  getExtraInfo: function(item) {
    // data binding can't handle spaces
    return item['extra info'];
  }
});
```

Now create /src/html/elements/table-example/css/table-detail.scss

/src/html/elements/table-example/css/table-detail.scss

```
:host {
  display: block;
  height: 8em;
}

paper-material {
  padding: 1em;
  margin: 1em;
}
```

The result will look like this:



1.2.9 Demo 6: Charts

For a complete list of options available for each type of chart, look at the official NVD3 docs (https://nvd3-community.github.io/nvd3/examples/documentation.html)

Make the chart-examples element

In your cell, run:

```
cd src/html/elements
./new-element.js
name of the new element: chart-examples
creating new element <chart-examples>...
removing any .svn folders in  chart-examples
Finished
```

Register the chart-examples element

Edit src/html/elements/elements.html and add the following line

```
<link rel="import" href="chart-examples/chart-examples.html">
```

This will tell AjaXell to load our new element.

Edit the chart-examples element

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/ts/common-elements/iron-flex-layout-attributes/iron-flex-
↪layout-attributes.html">
<link rel="import" href="/ts/common-elements/math-equation/math-equation.html">
<link rel="import" href="/ts/common-elements/charts/line-chart/line-chart.html">
<link rel="import" href="/ts/common-elements/charts/cumulative-line-chart/cumulative-
↪line-chart.html">
<link rel="import" href="/ts/common-elements/charts/focus-line-chart/focus-line-chart.
↪html">
<link rel="import" href="/ts/common-elements/charts/horizontal-stacked-bar-chart/
↪horizontal-stacked-bar-chart.html">
<link rel="import" href="/ts/common-elements/charts/historical-bar-chart/historical-
↪bar-chart.html">
<link rel="import" href="/ts/common-elements/charts/discrete-bar-chart/discrete-bar-
↪chart.html">
<link rel="import" href="/ts/common-elements/charts/stacked-bar-chart/stacked-bar-
↪chart.html">
<link rel="import" href="/ts/common-elements/charts/pie-chart/pie-chart.html">
<link rel="import" href="/ts/common-elements/charts/scatter-chart/scatter-chart.html">
<link rel="import" href="/ts/common-elements/charts/stacked-area-chart/stacked-area-
↪chart.html">
<link rel="import" href="/ts/common-elements/charts/parallel-chart/parallel-chart.html
↪">
<link rel="import" href="/ts/common-elements/charts/candlestick-chart/candlestick-
↪chart.html">

<!--
`chart-exampless` gives some examples of how to use chart elements in the
common-elements package.
-->
```

```
<dom-module id="chart-exampless">
  <template>
    <style include="reset-css"></style>
    <style include="iron-flex-layout-attributes"></style>
    <link rel="stylesheet" type="text/css" href="css/chart-exampless-min.css?__
 ↪inline=true">

  </template>
  <script src="javascript/chart-exampless-min.js?__inline=true"></script>
</dom-module>
```

We'll add more stuff as we go along.

Now edit /src/html/elements/chart-examples/css/chart-examples.scss

/src/html/elements/chart-examples/css/chart-examples.scss

```
:host {
  display: block;
  height: 100%;
}

paper-material {
  height: 400px;
  margin: 1em;
}
```

Now execute Grunt to build our new Polymer element.

```
cd src/html
grunt
```

Make the chart-examples panel

Make a new c++ file /src/common/panels/ChartExamples.cc

/src/common/panels/ChartExamples.cc

```
#include "subsystem/supervisor/panels/ChartExamples.h"
#include "ajax/PolymerElement.h"
#include "json/json.h"
#include <math.h>

using namespace subsystempanels;
ChartExamples::ChartExamples( tsframework::CellAbstractContext* context, log4cplus::
 ↪Logger& logger)
:tsframework::CellPanel(context,logger) {
  logger_ = log4cplus::Logger::getInstance(logger.getName() +".ChartExamples");
}

void ChartExamples::layout(cgicc::Cgicc& cgi) {
  remove();
  add(new ajax::PolymerElement("chart-examples"));
}
```

Make the /include/subsystem/supervisor/panels/ChartExamples.h file.

/include/subsystem/supervisor/panels/ChartExamples.h

```
#ifndef _subsystem_supervisor_panels_ChartExamples_h_
#define _subsystem_supervisor_panels_ChartExamples_h_

#include "ts/framework/CellPanel.h"
#include "log4cplus/logger.h"
#include "cgicc/Cgicc.h"

namespace subsystempanels {
  class ChartExamples: public tsframework::CellPanel {
    public:
      ChartExamples(tsframework::CellAbstractContext* context, log4cplus::Logger&
↪logger);
      void layout(cgicc::Cgicc& cgi);
    private:
  };
}
#endif
```

Register the new class in the Makefile.

```
Sources=\
    version.cc\
    Cell.cc\
    CellContext.cc\
    Configuration.cc\
    ...
    panels/ChartExamples.cc \
    ...
```

Now register your new panel in the menu so users can access it.

Edit src/common/Cell.cc

```
#include "subsystem/supervisor/panels/ChartExamples.h"
...
void subsystemsupervisor::Cell::init() {
  ...
  tsframework::CellPanelFactory* panelF = getContext()->getPanelFactory();
  ...
  panelF->add<subsystempanels::ChartExamples>("Chart examples");
```

Now you can compile your cell and you should see the "Form example" panel in the menu under the 'control-panels' section.

Chart1: A line chart

This example will make a basic line chart, and will show a simple line. Data is rendered client-side for simplicity.

Also note the use of the *math-equation* element do display LaTeX math.

HTML

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<dom-module id="chart-examples">
  <template>

    <div horizontal layout>
      <paper-material elevation="1" flex vertical layout>
        <math-equation big>f(x) = x^5 + 3.5x^4 - 2.4x^3 -12.5x^2 + 1.5x + 9</math-equation>
        <line-chart flex data="{{chart1data}}" config="{{chart1config}}" configure-chart="{{chart1JSConfig}}"></line-chart>
      </paper-material>
    </div>

  </template>
</dom-module>
```

JavaScript

For this first example, we generate our data client side.

Edit /src/html/elements/chart-examples/javascript/chart-examples.js

/src/html/elements/chart-examples/javascript/chart-examples.js

```
Polymer({
  is: 'chart-examples',
  properties: {
    chart1data: {
      type: Array,
      value: function() {
        var data = []
        for (var i = -3.5; i < 2.2; i = i+0.01) {
          data.push({
            x: i,
            y: Math.pow(i,5) + 3.5*Math.pow(i,4) - 2.5*Math.pow(i,3) - 12.5*Math.pow(i,2) + 1.5*i + 9
          });
        }

        return [
          {
            values: data,
            key: 'fifth degree polynomial',
            color: '#00671A'
          }
        ];
      }
    },

    chart1config: {
      type: Object,
      value: function() {
        return {
          useInteractiveGuideline: true,
          margin: {
            left: 70
          }
        }
```

```
      }
    },

    chart1JSConfig: {
      type: Function,
      value: function() {
        return function() {
          var superscript = "$^0$$^1$$^2$$^3$$^4$$^5$$^6$$^7$$^8$$^9$";
          var formatPower = function(d) {
            return (d + "").split("").map(function(c) {
              return superscript[c];
            }).join("");
          };
          this._chart.xAxis
            .axisLabel("x")
            .tickFormat(d3.format(',.2f'));

          this._chart.yAxis
            .axisLabel("x" + formatPower(5) + " + 3.5x" + formatPower(4) + " - 2.5x"
↪+ formatPower(3) + " - 12.5x" + formatPower(2) + " + 1.5x + 9")
            .tickFormat(d3.format(',.2f'));
        }
      }
    }
  }
});
```

The result will look like this:

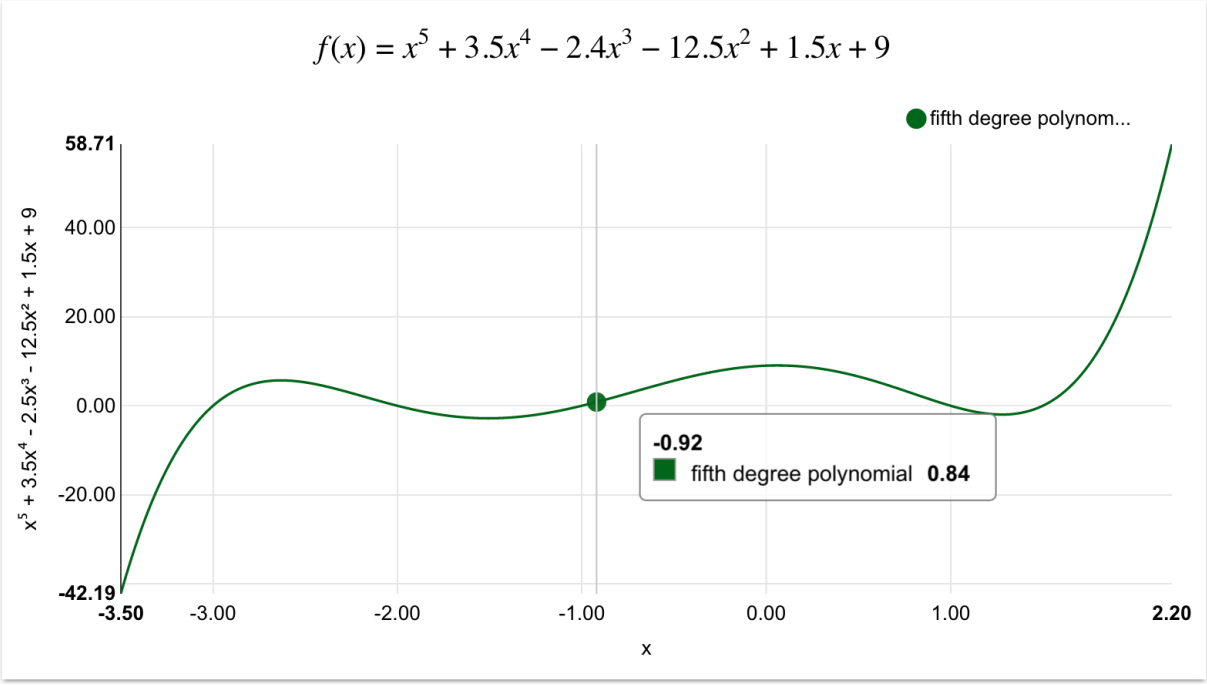


Chart2: A more advanced line chart

This line chart will feature multiple plots, missing data gaps, and a date for the x axis.

Data is still generated client side.

HTML & Javascript

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<dom-module id="chart-examples">
  <template>

    <div horizontal layout>
      <paper-material elevation="1" flex vertical layout>
        <math-equation big>f(x) = x^5 + 3.5x^4 - 2.4x^3 -12.5x^2 + 1.5x + 9</math-equation>
        <line-chart flex data="{{chart1data}}" config="{{chart1config}}" configure-chart="{{chart1JSConfig}}"></line-chart>
      </paper-material>
      <paper-material elevation="1" flex horizontal layout>
        <line-chart flex data="{{chart2data}}" config="{{chart1config}}" configure-chart="{{chart2JSConfig}}"></line-chart>
      </paper-material>
    </div>

  </template>
</dom-module>
```

Note that we put the second chart next to the first chart, this to demonstrate the ability to use the layout attributes.

Now edit /src/html/elements/chart-examples/javascript/chart-examples.js

/src/html/elements/chart-examples/javascript/chart-examples.js

```
Polymer({
  is: 'chart-examples',
  properties: {
    chart2data: {
      type: Array,
      value: function() {
        var sin = [],
            sin2 = [],
            cos = [],
            rand = [],
            rand2 = []
            ;
        for (var i = 0; i < 100; i++) {
            sin.push({x: i, y: i % 10 == 5 ? null : Math.sin(i/10) }); //the nulls are to show how defined works
            sin2.push({x: i, y: Math.sin(i/5) * 0.4 - 0.25});
            cos.push({x: i, y: .5 * Math.cos(i/10)});
            rand.push({x:i, y: Math.random() / 10});
            rand2.push({x: i, y: Math.cos(i/10) + Math.random() / 10 })
        }
        return [
            {
                area: true,
                values: sin,
                key: "Sine Wave",
```

```
                color: "#ff7f0e",
                strokeWidth: 4,
                classed: 'dashed'
            },
            {
                values: cos,
                key: "Cosine Wave",
                color: "#2ca02c"
            },
            {
                values: rand,
                key: "Random Points",
                color: "#2222ff"
            },
            {
                values: rand2,
                key: "Random Cosine",
                color: "#667711",
                strokeWidth: 3.5
            },
            {
                area: true,
                values: sin2,
                key: "Fill opacity",
                color: "#EF9CFB",
                fillOpacity: .1
            }
        ];
      },

    },

    chart2JSConfig: {
      type: Function,
      value: function() {
        return function() {
          this._chart.xAxis
            .showMaxMin(false)
            .tickFormat(function(d) { return d3.time.format('%x')(new Date(d)) });

          this._chart.yAxis
            .tickFormat(d3.format(',.2f'));
        }
      }
    }
  }
});
```

The result will look like this:

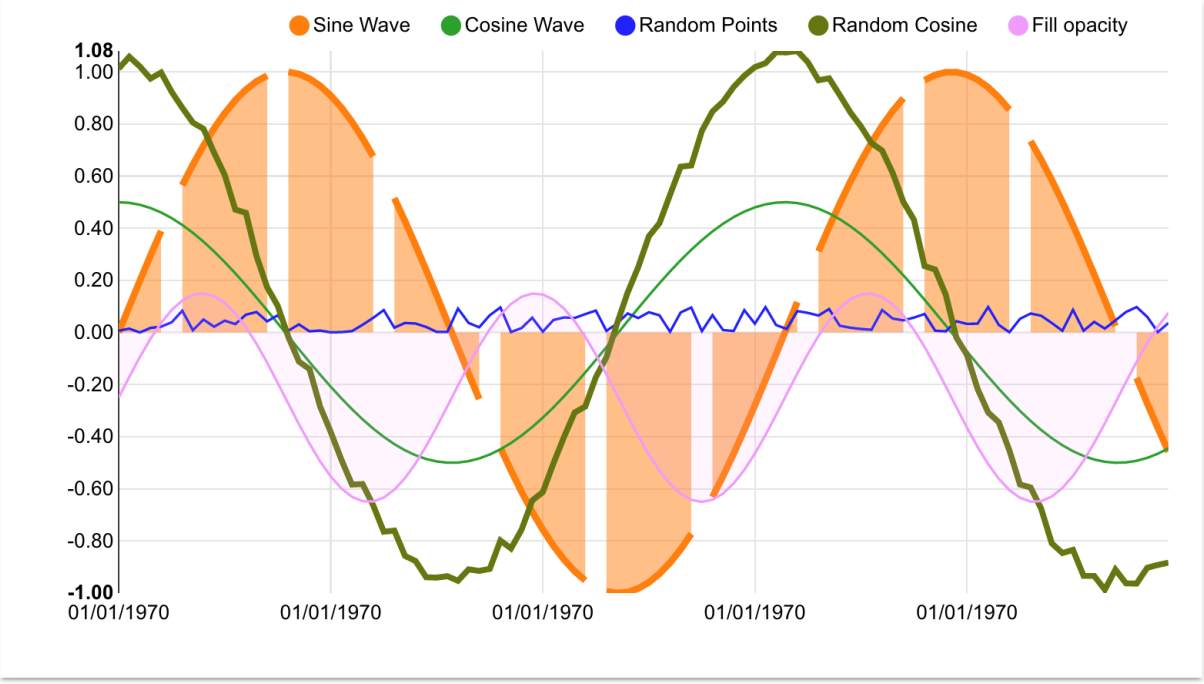


Chart3: A cumulative line chart

This type of chart shows a relative change of data. You will notice the data in this chart starts from 0. The user can also click on any point to instruct the chart to take that point in the x axis as the new relative zero.

HTML & Javascript

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<dom-module id="chart-examples">
  <template>

    <div horizontal layout>
      <paper-material elevation="1" flex horizontal layout>
        <cumulative-line-chart flex data="{{chart3data}}" config="{{chart3config}}" configure-chart="{{chart3JSConfig}}"></cumulative-line-chart>
      </div>
    </div>

  </template>
</dom-module>
```

Now edit /src/html/elements/chart-examples/javascript/chart-examples.js

/src/html/elements/chart-examples/javascript/chart-examples.js

```
Polymer({
  properties: {
    chart3data: {
```

```
      type: Array,
      value: function() {
        return [
          {
              key: "Long",
              values: [ [ 1083297600000 , -2.974623048543] , [ 1085976000000 , -1.
↪7740300785979] , [ 1088568000000 , 4.4681318138177] , [ 1091246400000 , 7.
↪0242541001353] , [ 1093924800000 , 7.5709603667586] , [ 1096516800000 , 20.
↪612245065736] , [ 1099195200000 , 21.698065237316] , [ 1101790800000 , 40.
↪501189458018] , [ 1104469200000 , 50.464679413194] , [ 1107147600000 , 48.
↪917421973355] , [ 1109566800000 , 63.750936549160] , [ 1112245200000 , 59.
↪072499126460] , [ 1114833600000 , 43.373158880492] , [ 1117512000000 , 54.
↪490918947556] , [ 1120104000000 , 56.661178852079] , [ 1122782400000 , 73.
↪450103545496] , [ 1125460800000 , 71.714526354907] , [ 1128052800000 , 85.
↪221664349607] , [ 1130734800000 , 77.769261392481] , [ 1133326800000 , 95.
↪966528716500] , [ 1136005200000 , 107.59132116397] , [ 1138683600000 , 127.
↪25740096723] , [ 1141102800000 , 122.13917498830] , [ 1143781200000 , 126.
↪53657279774] , [ 1146369600000 , 132.39300992970] , [ 1149048000000 , 120.
↪11238242904] , [ 1151640000000 , 118.41408917750] , [ 1154318400000 , 107.
↪92918924621] , [ 1156996800000 , 110.28057249569] , [ 1159588800000 , 117.
↪20485334692] , [ 1162270800000 , 141.33556756948] , [ 1164862800000 , 159.
↪59452727893] , [ 1167541200000 , 167.09801853304] , [ 1170219600000 , 185.
↪46849659215] , [ 1172638800000 , 184.82474099990] , [ 1175313600000 , 195.
↪63155213887] , [ 1177905600000 , 207.40597044171] , [ 1180584000000 , 230.
↪55966698196] , [ 1183176000000 , 239.55649035292] , [ 1185854400000 , 241.
↪35915085208] , [ 1188532800000 , 239.89428956243] , [ 1191124800000 , 260.
↪47781917715] , [ 1193803200000 , 276.39457482225] , [ 1196398800000 , 258.
↪66530682672] , [ 1199077200000 , 250.98846121893] , [ 1201755600000 , 226.
↪89902618127] , [ 1204261200000 , 227.29009273807] , [ 1206936000000 , 218.
↪66476654350] , [ 1209528000000 , 232.46605902918] , [ 1212206400000 , 253.
↪25667081117] , [ 1214798400000 , 235.82505363925] , [ 1217476800000 , 229.
↪70112774254] , [ 1220155200000 , 225.18472705952] , [ 1222747200000 , 189.
↪13661746552] , [ 1225425600000 , 149.46533007301] , [ 1228021200000 , 131.
↪00340772114] , [ 1230699600000 , 135.18341728866] , [ 1233378000000 , 109.
↪15296887173] , [ 1235797200000 , 84.614772549760] , [ 1238472000000 , 100.
↪60810015326] , [ 1241064000000 , 141.50134895610] , [ 1243742400000 , 142.
↪50405083675] , [ 1246334400000 , 139.81192372672] , [ 1249012800000 , 177.
↪78205544583] , [ 1251691200000 , 194.73691933074] , [ 1254283200000 , 209.
↪00838460225] , [ 1256961600000 , 198.19855877420] , [ 1259557200000 , 222.
↪37102417812] , [ 1262235600000 , 234.24581081250] , [ 1264914000000 , 228.
↪26087689346] , [ 1267333200000 , 248.81895126250] , [ 1270008000000 , 270.
↪57301075186] , [ 1272600000000 , 292.64604322550] , [ 1275278400000 , 265.
↪94088520518] , [ 1277870400000 , 237.82887467569] , [ 1280548800000 , 265.
↪55973314204] , [ 1283227200000 , 248.30877330928] , [ 1285819200000 , 278.
↪14870066912] , [ 1288497600000 , 292.69260960288] , [ 1291093200000 , 300.
↪84263809599] , [ 1293771600000 , 326.17253914628] , [ 1296450000000 , 337.
↪69335966505] , [ 1298869200000 , 339.73260965121] , [ 1301544000000 , 346.
↪87865120765] , [ 1304136000000 , 347.92991526628] , [ 1306814400000 , 342.
↪04627502669] , [ 1309406400000 , 333.45386231233] , [ 1312084800000 , 323.
↪15034181243] , [ 1314763200000 , 295.66126882331] , [ 1317355200000 , 251.
↪48014579253] , [ 1320033600000 , 295.15424257905] , [ 1322629200000 , 294.
↪54766764397] , [ 1325307600000 , 295.72906119051] , [ 1327986000000 , 325.
↪73351347613] , [ 1330491600000 , 340.16106061186] , [ 1333166400000 , 345.
↪15514071490] , [ 1335758400000 , 337.10259395679] , [ 1338436800000 , 318.
↪68216333837] , [ 1341028800000 , 317.03683945246] , [ 1343707200000 , 318.
↪53549659997] , [ 1346385600000 , 332.85381464104] , [ 1348977600000 , 337.
↪36534373477] , [ 1351656000000 , 350.27872156161] , [ 1354251600000 , 349.
↪45128876100]]
```

```
            ,
            mean: 250
        },
        {
            key: "Short",
            values: [ [ 1083297600000 , -0.77078283705125] , [ 1085976000000 , -1.
↪8356366650335] , [ 1088568000000 , -5.3121322073127] , [ 1091246400000 , -4.
↪9320975829662] , [ 1093924800000 , -3.9835408823225] , [ 1096516800000 , -6.
↪8694685316805] , [ 1099195200000 , -8.4854877428545] , [ 1101790800000 , -15.
↪933627197384] , [ 1104469200000 , -15.920980069544] , [ 1107147600000 , -12.
↪478685045651] , [ 1109566800000 , -17.297761889305] , [ 1112245200000 , -15.
↪247129891020] , [ 1114833600000 , -11.336459046839] , [ 1117512000000 , -13.
↪298990907415] , [ 1120104000000 , -16.360027000056] , [ 1122782400000 , -18.
↪527929522030] , [ 1125460800000 , -22.176516738685] , [ 1128052800000 , -23.
↪309665368330] , [ 1130734800000 , -21.629973409748] , [ 1133326800000 , -24.
↪186429093486] , [ 1136005200000 , -29.116707312531] , [ 1138683600000 , -37.
↪188037874864] , [ 1141102800000 , -34.689264821198] , [ 1143781200000 , -39.
↪505932105359] , [ 1146369600000 , -45.339572492759] , [ 1149048000000 , -43.
↪849353192764] , [ 1151640000000 , -45.418353922571] , [ 1154318400000 , -44.
↪579281059919] , [ 1156996800000 , -44.027098363370] , [ 1159588800000 , -41.
↪261306759439] , [ 1162270800000 , -47.446018534027] , [ 1164862800000 , -53.
↪413782948909] , [ 1167541200000 , -50.700723647419] , [ 1170219600000 , -56.
↪374090913296] , [ 1172638800000 , -61.754245220322] , [ 1175313600000 , -66.
↪246241587629] , [ 1177905600000 , -75.351650899999] , [ 1180584000000 , -81.
↪699058262032] , [ 1183176000000 , -82.487023368081] , [ 1185854400000 , -86.
↪230055113277] , [ 1188532800000 , -84.746914818507] , [ 1191124800000 , -100.
↪77134971977] , [ 1193803200000 , -109.95435565947] , [ 1196398800000 , -99.
↪605672965057] , [ 1199077200000 , -99.607249394382] , [ 1201755600000 , -94.
↪874614950188] , [ 1204261200000 , -105.35899063105] , [ 1206936000000 , -106.
↪01931193802] , [ 1209528000000 , -110.28883571771] , [ 1212206400000 , -119.
↪60256203030] , [ 1214798400000 , -115.62201315802] , [ 1217476800000 , -106.
↪63824185202] , [ 1220155200000 , -99.848746318951] , [ 1222747200000 , -85.
↪631219602987] , [ 1225425600000 , -63.547909262067] , [ 1228021200000 , -59.
↪753275364457] , [ 1230699600000 , -63.874977883542] , [ 1233378000000 , -56.
↪865697387488] , [ 1235797200000 , -54.285579501988] , [ 1238472000000 , -56.
↪474659581885] , [ 1241064000000 , -63.847137745644] , [ 1243742400000 , -68.
↪754247867325] , [ 1246334400000 , -69.474257009155] , [ 1249012800000 , -75.
↪084828197067] , [ 1251691200000 , -77.101028237237] , [ 1254283200000 , -80.
↪454866854387] , [ 1256961600000 , -78.984349952220] , [ 1259557200000 , -83.
↪041230807854] , [ 1262235600000 , -84.529748348935] , [ 1264914000000 , -83.
↪837470195508] , [ 1267333200000 , -87.174487671969] , [ 1270008000000 , -90.
↪342293007487] , [ 1272600000000 , -93.550928464991] , [ 1275278400000 , -85.
↪833102140765] , [ 1277870400000 , -79.326501831592] , [ 1280548800000 , -87.
↪986196903537] , [ 1283227200000 , -85.397862121771] , [ 1285819200000 , -94.
↪738167050020] , [ 1288497600000 , -98.661952897151] , [ 1291093200000 , -99.
↪609665952708] , [ 1293771600000 , -103.57099836183] , [ 1296450000000 , -104.
↪04353411322] , [ 1298869200000 , -108.21382792587] , [ 1301544000000 , -108.
↪74006900920] , [ 1304136000000 , -112.07766650960] , [ 1306814400000 , -109.
↪63328199118] , [ 1309406400000 , -106.53578966772] , [ 1312084800000 , -103.
↪16480871469] , [ 1314763200000 , -95.945078001828] , [ 1317355200000 , -81.
↪226687340874] , [ 1320033600000 , -90.782206596168] , [ 1322629200000 , -89.
↪484445370113] , [ 1325307600000 , -88.514723135326] , [ 1327986000000 , -93.
↪381292724320] , [ 1330491600000 , -97.529705609172] , [ 1333166400000 , -99.
↪520481439189] , [ 1335758400000 , -99.430184898669] , [ 1338436800000 , -93.
↪349934521973] , [ 1341028800000 , -95.858475286491] , [ 1343707200000 , -95.
↪522755836605] , [ 1346385600000 , -98.503848862036] , [ 1348977600000 , -101.
↪49415251896] , [ 1351656000000 , -101.50099325672] , [ 1354251600000 , -99.
↪487094927489]]
```

```
16              ,
17              mean: -60
18          },
19          {
20              key: "Gross",
21              mean: 125,
22              values: [ [ 1083297600000 , -3.7454058855943] , [ 1085976000000 , -3.
↪6096667436314] , [ 1088568000000 , -0.8440003934950] , [ 1091246400000 , 2.
↪0921565171691] , [ 1093924800000 , 3.5874194844361] , [ 1096516800000 , 13.
↪742776534056] , [ 1099195200000 , 13.212577494462] , [ 1101790800000 , 24.
↪567562260634] , [ 1104469200000 , 34.543699343650] , [ 1107147600000 , 36.
↪438736927704] , [ 1109566800000 , 46.453174659855] , [ 1112245200000 , 43.
↪825369235440] , [ 1114833600000 , 32.036699833653] , [ 1117512000000 , 41.
↪191928040141] , [ 1120104000000 , 40.301151852023] , [ 1122782400000 , 54.
↪922174023466] , [ 1125460800000 , 49.538009616222] , [ 1128052800000 , 61.
↪911998981277] , [ 1130734800000 , 56.139287982733] , [ 1133326800000 , 71.
↪780099623014] , [ 1136005200000 , 78.474613851439] , [ 1138683600000 , 90.
↪069363092366] , [ 1141102800000 , 87.449910167102] , [ 1143781200000 , 87.
↪030640692381] , [ 1146369600000 , 87.053437436941] , [ 1149048000000 , 76.
↪263029236276] , [ 1151640000000 , 72.995735254929] , [ 1154318400000 , 63.
↪349908186291] , [ 1156996800000 , 66.253474132320] , [ 1159588800000 , 75.
↪943546587481] , [ 1162270800000 , 93.889549035453] , [ 1164862800000 , 106.
↪18074433002] , [ 1167541200000 , 116.39729488562] , [ 1170219600000 , 129.
↪09440567885] , [ 1172638800000 , 123.07049577958] , [ 1175313600000 , 129.
↪38531055124] , [ 1177905600000 , 132.05431954171] , [ 1180584000000 , 148.
↪86060871993] , [ 1183176000000 , 157.06946698484] , [ 1185854400000 , 155.
↪12909573880] , [ 1188532800000 , 155.14737474392] , [ 1191124800000 , 159.
↪70646945738] , [ 1193803200000 , 166.44021916278] , [ 1196398800000 , 159.
↪05963386166] , [ 1199077200000 , 151.38121182455] , [ 1201755600000 , 132.
↪02441123108] , [ 1204261200000 , 121.93110210702] , [ 1206936000000 , 112.
↪64545460548] , [ 1209528000000 , 122.17722331147] , [ 1212206400000 , 133.
↪65410878087] , [ 1214798400000 , 120.20304048123] , [ 1217476800000 , 123.
↪06288589052] , [ 1220155200000 , 125.33598074057] , [ 1222747200000 , 103.
↪50539786253] , [ 1225425600000 , 85.917420810943] , [ 1228021200000 , 71.
↪250132356683] , [ 1230699600000 , 71.308439405118] , [ 1233378000000 , 52.
↪287271484242] , [ 1235797200000 , 30.329193047772] , [ 1238472000000 , 44.
↪133440571375] , [ 1241064000000 , 77.654211210456] , [ 1243742400000 , 73.
↪749802969425] , [ 1246334400000 , 70.337666717565] , [ 1249012800000 , 102.
↪69722724876] , [ 1251691200000 , 117.63589109350] , [ 1254283200000 , 128.
↪55351774786] , [ 1256961600000 , 119.21420882198] , [ 1259557200000 , 139.
↪32979337027] , [ 1262235600000 , 149.71606246357] , [ 1264914000000 , 144.
↪42340669795] , [ 1267333200000 , 161.64446359053] , [ 1270008000000 , 180.
↪23071774437] , [ 1272600000000 , 199.09511476051] , [ 1275278400000 , 180.
↪10778306442] , [ 1277870400000 , 158.50237284410] , [ 1280548800000 , 177.
↪57353623850] , [ 1283227200000 , 162.91091118751] , [ 1285819200000 , 183.
↪41053361910] , [ 1288497600000 , 194.03065670573] , [ 1291093200000 , 201.
↪23297214328] , [ 1293771600000 , 222.60154078445] , [ 1296450000000 , 233.
↪35556801977] , [ 1298869200000 , 231.22452435045] , [ 1301544000000 , 237.
↪84432503045] , [ 1304136000000 , 235.55799131184] , [ 1306814400000 , 232.
↪11873570751] , [ 1309406400000 , 226.62381538123] , [ 1312084800000 , 219.
↪34811113539] , [ 1314763200000 , 198.69242285581] , [ 1317355200000 , 168.
↪90235629066] , [ 1320033600000 , 202.64725756733] , [ 1322629200000 , 203.
↪05389378105] , [ 1325307600000 , 204.85986680865] , [ 1327986000000 , 229.
↪77085616585] , [ 1330491600000 , 239.65202435959] , [ 1333166400000 , 242.
↪33012622734] , [ 1335758400000 , 234.11773262149] , [ 1338436800000 , 221.
↪47846307887] , [ 1341028800000 , 216.98308827912] , [ 1343707200000 , 218.
↪37781386755] , [ 1346385600000 , 229.39368622736] , [ 1348977600000 , 230.
↪54656412916] , [ 1351656000000 , 243.06087025523] , [ 1354251600000 , 244.
↪24733578385]]
```

```
23          },
24          {
25              key: "S&P 1500",
26              values: [ [ 1083297600000 , -1.7798428181819] , [ 1085976000000 , -0.
↪36883324836999] , [ 1088568000000 , 1.7312581046040] , [ 1091246400000 , -1.
↪8356125950460] , [ 1093924800000 , -1.5396564170877] , [ 1096516800000 , -0.
↪16867791409247] , [ 1099195200000 , 1.3754263993413] , [ 1101790800000 , 5.
↪8171640898041] , [ 1104469200000 , 9.4350145241608] , [ 1107147600000 , 6.
↪7649081510160] , [ 1109566800000 , 9.1568499314776] , [ 1112245200000 , 7.
↪2485090994419] , [ 1114833600000 , 4.8762222306595] , [ 1117512000000 , 8.
↪5992339354652] , [ 1120104000000 , 9.0896517982086] , [ 1122782400000 , 13.
↪394644048577] , [ 1125460800000 , 12.311842010760] , [ 1128052800000 , 13.
↪221003650717] , [ 1130734800000 , 11.218481009206] , [ 1133326800000 , 15.
↪565352598445] , [ 1136005200000 , 15.623703865926] , [ 1138683600000 , 19.
↪275255326383] , [ 1141102800000 , 19.432433717836] , [ 1143781200000 , 21.
↪232881244655] , [ 1146369600000 , 22.798299192958] , [ 1149048000000 , 19.
↪006125095476] , [ 1151640000000 , 19.151889158536] , [ 1154318400000 , 19.
↪340022855452] , [ 1156996800000 , 22.027934841859] , [ 1159588800000 , 24.
↪903300681329] , [ 1162270800000 , 29.146492833877] , [ 1164862800000 , 31.
↪781626082589] , [ 1167541200000 , 33.358770738428] , [ 1170219600000 , 35.
↪622684613497] , [ 1172638800000 , 33.332821711366] , [ 1175313600000 , 34.
↪878748635832] , [ 1177905600000 , 40.582332613844] , [ 1180584000000 , 45.
↪719535502920] , [ 1183176000000 , 43.239344722386] , [ 1185854400000 , 38.
↪550955100342] , [ 1188532800000 , 40.585368816283] , [ 1191124800000 , 45.
↪601374057981] , [ 1193803200000 , 48.051404337892] , [ 1196398800000 , 41.
↪582581696032] , [ 1199077200000 , 40.650580792748] , [ 1201755600000 , 32.
↪252222066493] , [ 1204261200000 , 28.106390258553] , [ 1206936000000 , 27.
↪532698196687] , [ 1209528000000 , 33.986390463852] , [ 1212206400000 , 36.
↪302660526438] , [ 1214798400000 , 25.015574480172] , [ 1217476800000 , 23.
↪989494069029] , [ 1220155200000 , 25.934351445531] , [ 1222747200000 , 14.
↪627592011699] , [ 1225425600000 , -5.2249403809749] , [ 1228021200000 , -12.
↪330933408050] , [ 1230699600000 , -11.000291508188] , [ 1233378000000 , -18.
↪563864948088] , [ 1235797200000 , -27.213097001687] , [ 1238472000000 , -20.
↪834133840523] , [ 1241064000000 , -12.717886701719] , [ 1243742400000 , -8.
↪1644613083526] , [ 1246334400000 , -7.9108408918201] , [ 1249012800000 , -0.
↪77002391591209] , [ 1251691200000 , 2.8243816569672] , [ 1254283200000 , 6.
↪8761411421070] , [ 1256961600000 , 4.5060912230294] , [ 1259557200000 , 10.
↪487179794349] , [ 1262235600000 , 13.251375597594] , [ 1264914000000 , 9.
↪2207594803415] , [ 1267333200000 , 12.836276936538] , [ 1270008000000 , 19.
↪816793904978] , [ 1272600000000 , 22.156787167211] , [ 1275278400000 , 12.
↪518039090576] , [ 1277870400000 , 6.4253587440854] , [ 1280548800000 , 13.
↪847372028409] , [ 1283227200000 , 8.5454736090364] , [ 1285819200000 , 18.
↪542801953304] , [ 1288497600000 , 23.037064683183] , [ 1291093200000 , 23.
↪517422401888] , [ 1293771600000 , 31.804723416068] , [ 1296450000000 , 34.
↪778247386072] , [ 1298869200000 , 39.584883855230] , [ 1301544000000 , 40.
↪080647664875] , [ 1304136000000 , 44.180050667889] , [ 1306814400000 , 42.
↪533535927221] , [ 1309406400000 , 40.105374449011] , [ 1312084800000 , 37.
↪014659267156] , [ 1314763200000 , 29.263745084262] , [ 1317355200000 , 19.
↪637463417584] , [ 1320033600000 , 33.157645345770] , [ 1322629200000 , 32.
↪895053150988] , [ 1325307600000 , 34.111544824647] , [ 1327986000000 , 40.
↪453985817473] , [ 1330491600000 , 46.435700783313] , [ 1333166400000 , 51.
↪062385488671] , [ 1335758400000 , 50.130448220658] , [ 1338436800000 , 41.
↪035476682018] , [ 1341028800000 , 46.591932296457] , [ 1343707200000 , 48.
↪349391180634] , [ 1346385600000 , 51.913011286919] , [ 1348977600000 , 55.
↪747238313752] , [ 1351656000000 , 52.991824077209] , [ 1354251600000 , 49.
↪556311883284]]
27          }
28        ];
```

```
      }
    },

    chart3config: {
      type: Object,
      value: function() {
        return {
          showLegend : true,

          x: function(d) { return d[0] },
          y: function(d) { return d[1]/100 },
          color: d3.scale.category10().range(),
          average: function(d) { return d.mean/100; },
          duration: 300,
          clipVoronoi: false,
          useInteractiveGuideline: true,

          //showcase that the config is idiot-proof
          donut : true
        }
      }
    },

    chart3JSConfig: {
      type: Function,
      value: function() {
        return function() {
          this._chart.xAxis
            .tickFormat(function(d) { return d3.time.format('%m/%d/%y')(new Date(d)) }
↪);

          this._chart.yAxis
            .tickFormat(d3.format(',.1%'));
        }
      }
    }
  }
});
```

The result will look like this:

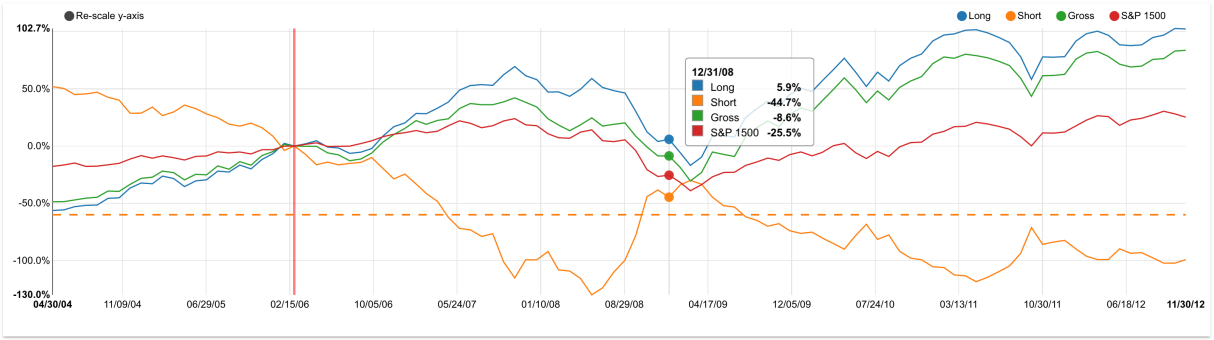


Chart4: A focusable line chart

This line chart features a bar on the bottom the user can use to zoom in or out of the dataset.

Also this chart has been configured with an upper limit on the y axis to prevent very high initial values from scaling down the rest of the chart.

This example will be the first to have data generated on server-side.

C++ code

Edit /src/common/panels/ChartExamples.cc

/src/common/panels/ChartExamples.cc

```
void ChartExamples::layout(cgicc::Cgicc& cgi) {
  remove();
  setEvent("getChart4", ajax::Eventable::OnClick, this, &ChartExamples::getChart4);
  add(new ajax::PolymerElement("chart-examples"));
}

void ChartExamples::getChart4(cgicc::Cgicc& cgi,std::ostream& out) {
  int pix = 0;
  Json::Value root(Json::arrayValue);

  Json::Value stream0;
  stream0["area"] = true;
  stream0["color"] = "#00695C";
  stream0["key"] = "prime density π(x)/x";
  Json::Value stream0values(Json::arrayValue);
  for (int i = 2; i < 200; i++) {
    if (ChartExamples::isPrime(i)) {
      pix += 1;
    }
    Json::Value item;
    item["x"] = i;
    item["y"] = (float)pix/(float)i;
    stream0values.append(item);
  }
  stream0["values"] = stream0values;
  root.append(stream0);

  Json::Value stream1;
  stream1["area"] = false;
  stream1["color"] = "#E57373";
  stream1["key"] = "1/(ln(x)-1)";
  Json::Value stream1values(Json::arrayValue);
  for (int i = 2; i < 200; i++) {
    Json::Value item;
    item["x"] = i;
    item["y"] = (float)1/(float)(log(i) - 1);
    stream1values.append(item);
  }
  stream1["values"] = stream1values;
  root.append(stream1);

  out << root;
}

bool ChartExamples::isPrime(int num) {
  if (num <= 1)
    return false;
```

```
  else if (num == 2)
    return true;
  else if (num % 2 == 0)
    return false;
  else {
    bool prime = true;
    int divisor = 3;
    double num_d = static_cast<double>(num);
    int upperLimit = static_cast<int>(sqrt(num_d) + 1);

    while (divisor <= upperLimit) {
      if (num % divisor == 0)
        prime = false;
      divisor += 2;
    }
    return prime;
  }
}
```

Edit /include/subsystem/supervisor/panels/ChartExamples.h.

/include/subsystem/supervisor/panels/ChartExamples.h

```
namespace subsystempanels {
  class ChartExamples: public tsframework::CellPanel {
    private:
      void getChart4(cgicc::Cgicc& cgi,std::ostream& out);
      bool isPrime(int num);
  };
}
```

HTML & Javascript

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<dom-module id="chart-examples">
  <template>

    <div horizontal layout>
      <paper-material elevation="1" flex vertical layout>
        <ts-ajax data="{{chart4data}}" callback="getChart4" handle-as="json" auto></
↪ts-ajax>
        <p>
          The number of primes with value lower than x ( <math-equation big inline>
↪\pi(x)</math-equation> ) can be approximated using <math-equation big inline>\frac
↪{1}{\ln(x) -1}</math-equation>
        </p>
        <focus-line-chart flex data="{{chart4data}}" config="{{chart4config}}"␣
↪configure-chart="{{chart4JSConfig}}"></focus-line-chart>
      </div>
    </div>

  </template>
</dom-module>
```

Now edit /src/html/elements/chart-examples/javascript/chart-examples.js

/src/html/elements/chart-examples/javascript/chart-examples.js

```
Polymer({
  properties: {
    chart4data: {
      type: Array,
      value: function() {
        return [];
      }
    },

    chart4config: {
      type: Object,
      value: function() {
        return {
          brushExtent: [2,100],
          useInteractiveGuideline: true
        }
      }
    },

    chart4JSConfig: {
      type: Function,
      value: function() {
        return function() {
          this._chart.xAxis
            .tickFormat(d3.format(',f'));
          this._chart.x2Axis
            .tickFormat(d3.format(',f'));

          this._chart.yAxis
            .tickFormat(d3.format(',.2f'));
          this._chart.y2Axis
            .tickFormat(d3.format(',.2f'));

          this._chart.yDomain([0.2,1]);
        }
      }
    }
  }
});
```

The result will look like this:

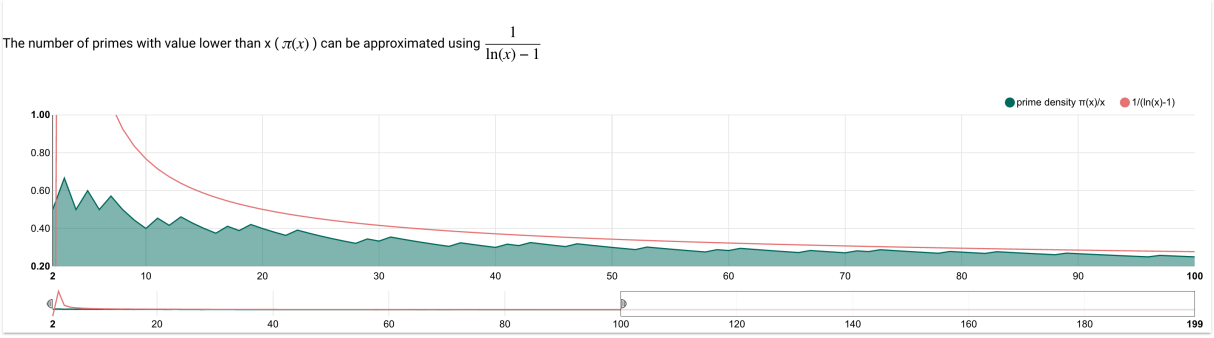


Chart5: Bar charts

For simplicity we will generate the data on client-side again.

In this example we will make a bar chart, a horizontal bar chart, a discrete bar chart, and a historical bar chart.

HTML & Javascript

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<dom-module id="chart-examples">
  <template>

    <div vertical layout>
      <div horizontal layout flex>
        <paper-material elevation="1" flex horizontal layout>
          <discrete-bar-chart flex data="{{chart5data}}" config="{{chart5config}}" configure-chart="{{chart5JSConfig}}"></discrete-bar-chart>
        </paper-material>
        <paper-material elevation="1" flex horizontal layout>
          <historical-bar-chart flex data="{{chart6data}}" config="{{chart6config}}" configure-chart="{{chart6JSConfig}}"></historical-bar-chart>
        </paper-material>
      </div>
      <div horizontal layout flex>
        <paper-material elevation="1" flex horizontal layout>
          <stacked-bar-chart flex data="{{chart5data}}" config="{{chart5config}}" configure-chart="{{chart5JSConfig}}"></stacked-bar-chart>
        </paper-material>
        <paper-material elevation="1" flex horizontal layout>
          <horizontal-stacked-bar-chart flex data="{{chart5data}}" config="{{chart5config}}" configure-chart="{{chart5JSConfig}}"></horizontal-stacked-bar-chart>
        </paper-material>
      </div>
    </div>

  </template>
</dom-module>
```

Now edit /src/html/elements/chart-examples/javascript/chart-examples.js

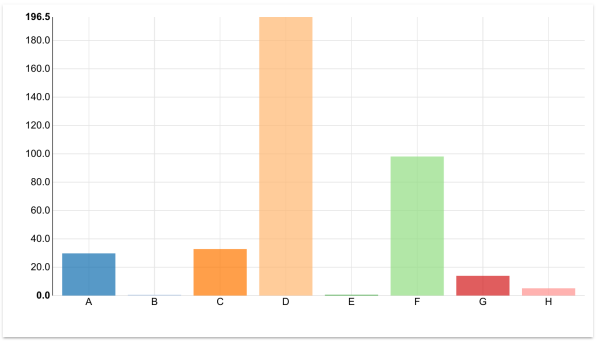
/src/html/elements/chart-examples/javascript/chart-examples.js

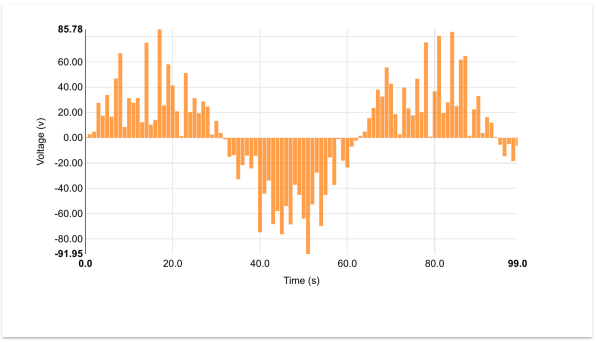
```
Polymer({
  properties: {
    chart5data: {
      type: Array,
      value: function() {
        return[{
          key: "Cumulative Return",
          values: [{
            "label": "A",
            "value": 29.765957771107
          }, {
            "label": "B",
```

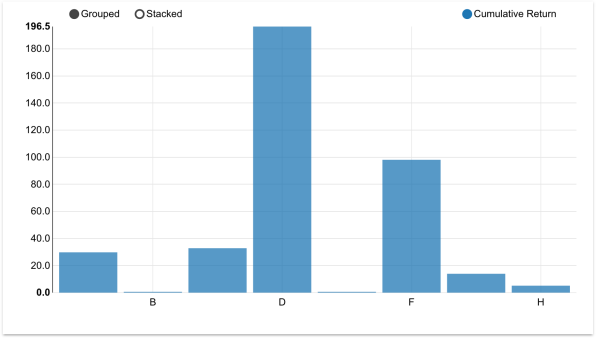
```
            "value": 0
          }, {
            "label": "C",
            "value": 32.807804682612
          }, {
            "label": "D",
            "value": 196.45946739256
          }, {
            "label": "E",
            "value": 0.19434030906893
          }, {
            "label": "F",
            "value": 98.079782601442
          }, {
            "label": "G",
            "value": 13.925743130903
          }, {
            "label": "H",
            "value": 5.1387322875705
          }]
        }];
      }
    },

    chart5config: {
      type: Object,
      value: function() {
        return {
          duration: 250,
          x: function(d) { return d.label },
          y: function(d) { return d.value },
        }
      }
    },

    chart5JSConfig: {
      type: Function,
      value: function() {
        return function() {
        }
      }
    },

    chart6data: {
      type: Array,
      value: function() {
        var sin = [];
        for (var i = 0; i < 100; i++) {
          sin.push({x: i, y: Math.sin(i/10) * Math.random() * 100});
        }
        return [{
          values: sin,
          key: "Sine Wave",
          color: "#ff7f0e"
        }];
      }
    },

```

```
    chart6config: {
      type: Object,
      value: function() {
        return {
          margin: {left: 100, bottom: 100},
          useInteractiveGuideline: true,
          duration: 250
        }
      }
    },

    chart6JSConfig: {
      type: Function,
      value: function() {
        return function() {
          this._chart.xAxis
            .axisLabel("Time (s)")
            .tickFormat(d3.format(',.1f'));
          this._chart.yAxis
            .axisLabel('Voltage (v)')
            .tickFormat(d3.format(',.2f'));
          this._chart.showXAxis(true);
        }
      }
    }
  }
});
```

The result will look like this:







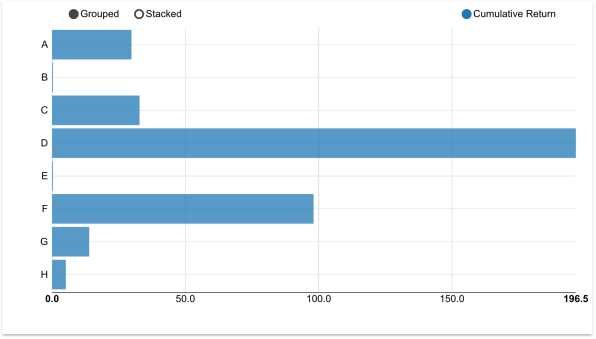


Chart6: Pie charts

HTML & Javascript

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<dom-module id="chart-examples">
  <template>

    <div vertical layout>
      <div horizontal layout flex>
        <paper-material elevation="1" flex horizontal layout>
          <pie-chart flex data="{{chart7data}}" config="{{chart7config}}"></pie-chart>
        </paper-material>
        <paper-material elevation="1" flex horizontal layout>
          <pie-chart flex data="{{chart7data}}" config="{{chart7config}}" configure-
↪chart="{{chart7JSConfig}}"></pie-chart>
        </paper-material>
      </div>
    </div>

  </template>
</dom-module>
```

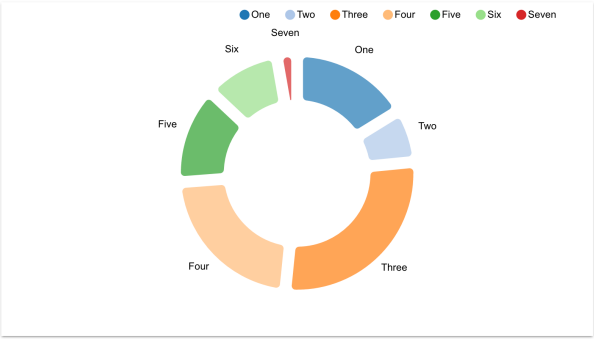
Now edit /src/html/elements/chart-examples/javascript/chart-examples.js

/src/html/elements/chart-examples/javascript/chart-examples.js

```
Polymer({
  properties: {
    chart7data: {
      type: Array,
      value: function() {
        var sin = [];
        for (var i = 0; i < 100; i++) {
          sin.push({x: i, y: Math.sin(i/10) * Math.random() * 100});
        }
        return [
          {key: "One", y: 5},
          {key: "Two", y: 2},
          {key: "Three", y: 9},
          {key: "Four", y: 7},
          {key: "Five", y: 4},
          {key: "Six", y: 3},
          {key: "Seven", y: 0.5}
        ];
      }
    },

    chart7config: {
      type: Object,
      value: function() {
        return {
          x: function(d) { return d.key },
          y: function(d) { return d.y },
          donut: true,
          padAngle: 0.08,
```

```
          cornerRadius: 5,
          labelsOutside: true
        }
      }
    },

    chart7JSConfig: {
      type: Function,
      value: function() {
        return function() {
          this._chart.pie
            .startAngle(function(d) { return d.startAngle/2 -Math.PI/2 })
            .endAngle(function(d) { return d.endAngle/2 -Math.PI/2 });
        }
      }
    }
  }
});
```

The result will look like this:



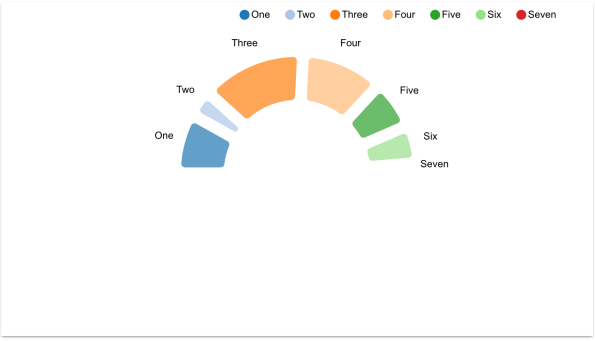


Chart7: Scatter charts

HTML & Javascript

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<dom-module id="chart-examples">
  <template>

    <div horizontal layout flex>
      <paper-material elevation="1" flex horizontal layout>
        <scatter-chart flex data="{{chart8data}}" config="{{chart8config}}" configure-
↪chart="{{chart8JSConfig}}"></scatter-chart>
      </paper-material>
    </div>

  </template>
</dom-module>
```

Now edit /src/html/elements/chart-examples/javascript/chart-examples.js

/src/html/elements/chart-examples/javascript/chart-examples.js

```
Polymer({
  properties: {
    chart8data: {
      type: Array,
      value: function() {
        var groups = 4;
        var points = 40;
        var data = [],
            shapes = ['thin-x', 'circle', 'cross', 'triangle-up', 'triangle-down',
↪'diamond', 'square'],
            random = d3.random.normal();
        for (i = 0; i < groups; i++) {
            data.push({
                key: 'Group ' + i,
                values: []
            });
            for (j = 0; j < points; j++) {
                data[i].values.push({
                    x: random(),
                    y: random(),
                    size: Math.round(Math.random() * 100) / 100,
                    shape: shapes[j % shapes.length]
                });
            }
        }
        return data;
      }
    },

    chart8config: {
      type: Object,
      value: function() {
        return {
          showDistX: true,
          showDistY: true,
          useVoronoi: true,
          duration: 300,
          color: d3.scale.category10().range()
        }
      }
    },

    chart8JSConfig: {
      type: Function,
      value: function() {
        return function() {
          nv.utils.symbolMap.set('thin-x', function(size) {
            size = Math.sqrt(size);
            return 'M' + (-size/2) + ',' + (-size/2) +
                    'l' + size + ',' + size +
                    'm0,' + -(size) +
                    'l' + (-size) + ',' + size;
          });
          this._chart.xAxis
            .tickFormat(d3.format('.02f'));
          this._chart.yAxis
            .tickFormat(d3.format('.02f'));
        }
```

```
        }
      }
    }
});
```

The result will look like this:

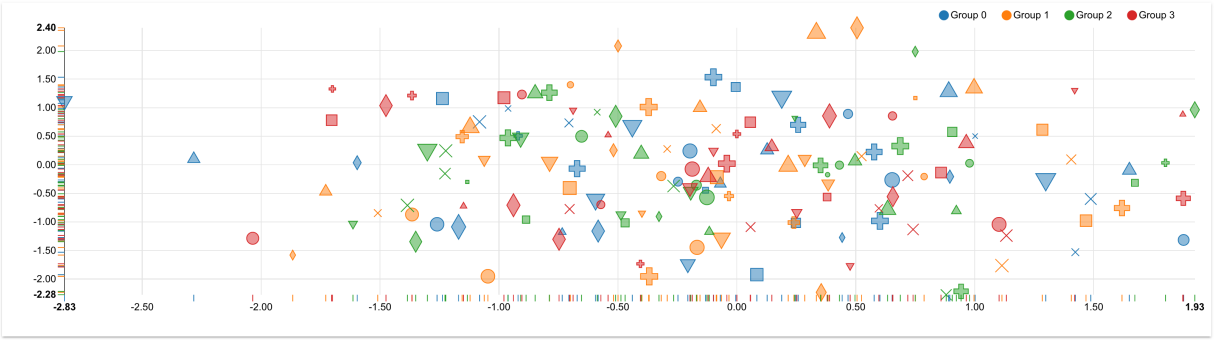


Chart8: Stacked area chart

HTML & Javascript

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<dom-module id="chart-examples">
  <template>

    <div horizontal layout flex>
      <paper-material elevation="1" flex horizontal layout>
        <stacked-area-chart flex data="{{chart9data}}" config="{{chart9config}}"␣
↪configure-chart="{{chart9JSConfig}}"></stacked-area-chart>
      </paper-material>
    </div>

  </template>
</dom-module>
```

Now edit /src/html/elements/chart-examples/javascript/chart-examples.js

/src/html/elements/chart-examples/javascript/chart-examples.js

```
Polymer({
  properties: {
    chart9data: {
      type: Array,
      value: function() {
        return [{
          "key": "Consumer Discretionary",
          "values": [
            [1138683600000, 27.38478809681],
            [1141102800000, 27.371377218208],
            [1143781200000, 26.309915460827],
            [1146369600000, 26.425199957521],
            [1149048000000, 26.823411519395],
```

```
            [1151640000000, 23.850443591584],
            [1154318400000, 23.158355444054],
            [1156996800000, 22.998689393694],
            [1159588800000, 27.977128511299],
            [1162270800000, 29.073672469721],
            [1164862800000, 28.587640408904],
            [1167541200000, 22.788453687638],
            [1170219600000, 22.429199073597],
            [1172638800000, 22.324103271051],
            [1175313600000, 17.558388444186],
            [1177905600000, 16.769518096208],
            [1180584000000, 16.214738201302],
            [1183176000000, 18.729632971228],
            [1185854400000, 18.814523318848],
            [1188532800000, 19.789986451358],
            [1191124800000, 17.070049054933],
            [1193803200000, 16.121349575715],
            [1196398800000, 15.141659430091],
            [1199077200000, 17.175388025298],
            [1201755600000, 17.286592443521],
            [1204261200000, 16.323141626569],
            [1206936000000, 19.231263773952],
            [1209528000000, 18.446256391094],
            [1212206400000, 17.822632399764],
            [1214798400000, 15.539366475979],
            [1217476800000, 15.255131790216],
            [1220155200000, 15.660963922593],
            [1222747200000, 13.254482273697],
            [1225425600000, 11.920796202299],
            [1228021200000, 12.122809090925],
            [1230699600000, 15.691026271393],
            [1233378000000, 14.720881635107],
            [1235797200000, 15.387939360044],
            [1238472000000, 13.765436672229],
            [1241064000000, 14.6314458648],
            [1243742400000, 14.292446536221],
            [1246334400000, 16.170071367016],
            [1249012800000, 15.948135554337],
            [1251691200000, 16.612872685134],
            [1254283200000, 18.778338719091],
            [1256961600000, 16.75602606542],
            [1259557200000, 19.385804443147],
            [1262235600000, 22.950590240168],
            [1264914000000, 23.61159018141],
            [1267333200000, 25.708586989581],
            [1270008000000, 26.883915999885],
            [1272600000000, 25.893486687065],
            [1275278400000, 24.678914263176],
            [1277870400000, 25.937275793023],
            [1280548800000, 29.46138169384],
            [1283227200000, 27.357322961862],
            [1285819200000, 29.057235285673],
            [1288497600000, 28.549434189386],
            [1291093200000, 28.506352379723],
            [1293771600000, 29.449241421597],
            [1296450000000, 25.796838168807],
            [1298869200000, 28.740145449189],
            [1301544000000, 22.091744141872],
```

```
            [1304136000000, 25.079662545409],
            [1306814400000, 23.674906973064],
            [1309406400000, 23.41800274293],
            [1312084800000, 23.243644138871],
            [1314763200000, 31.591854066817],
            [1317355200000, 31.497112374114],
            [1320033600000, 26.672380820431],
            [1322629200000, 27.297080015495],
            [1325307600000, 20.174315530051],
            [1327986000000, 19.631084213899],
            [1330491600000, 20.366462219462],
            [1333166400000, 17.429019937289],
            [1335758400000, 16.75543633539],
            [1338436800000, 16.182906906042]
          ]
        }, {
          "key": "Consumer Staples",
          "values": [
            [1138683600000, 7.2800122043237],
            [1141102800000, 7.1187787503354],
            [1143781200000, 8.351887016482],
            [1146369600000, 8.4156698763993],
            [1149048000000, 8.1673298604231],
            [1151640000000, 5.5132447126042],
            [1154318400000, 6.1152537710599],
            [1156996800000, 6.076765091942],
            [1159588800000, 4.6304473798646],
            [1162270800000, 4.6301068469402],
            [1164862800000, 4.3466656309389],
            [1167541200000, 6.830104897003],
            [1170219600000, 7.241633040029],
            [1172638800000, 7.1432372054153],
            [1175313600000, 10.608942063374],
            [1177905600000, 10.914964549494],
            [1180584000000, 10.933223880565],
            [1183176000000, 8.3457524851265],
            [1185854400000, 8.1078413081882],
            [1188532800000, 8.2697185922474],
            [1191124800000, 8.4742436475968],
            [1193803200000, 8.4994601179319],
            [1196398800000, 8.7387319683243],
            [1199077200000, 6.8829183612895],
            [1201755600000, 6.984133637885],
            [1204261200000, 7.0860136043287],
            [1206936000000, 4.3961787956053],
            [1209528000000, 3.8699674365231],
            [1212206400000, 3.6928925238305],
            [1214798400000, 6.7571718894253],
            [1217476800000, 6.4367313362344],
            [1220155200000, 6.4048441521454],
            [1222747200000, 5.4643833239669],
            [1225425600000, 5.3150786833374],
            [1228021200000, 5.3011272612576],
            [1230699600000, 4.1203601430809],
            [1233378000000, 4.0881783200525],
            [1235797200000, 4.1928665957189],
            [1238472000000, 7.0249415663205],
            [1241064000000, 7.006530880769],
```

```
            [1243742400000, 6.994835633224],
            [1246334400000, 6.1220222336254],
            [1249012800000, 6.1177436137653],
            [1251691200000, 6.1413396231981],
            [1254283200000, 4.8046006145874],
            [1256961600000, 4.6647600660544],
            [1259557200000, 4.544865006255],
            [1262235600000, 6.0488249316539],
            [1264914000000, 6.3188669540206],
            [1267333200000, 6.5873958262306],
            [1270008000000, 6.2281189839578],
            [1272600000000, 5.8948915746059],
            [1275278400000, 5.5967320482214],
            [1277870400000, 0.99784432084837],
            [1280548800000, 1.0950794175359],
            [1283227200000, 0.94479734407491],
            [1285819200000, 1.222093988688],
            [1288497600000, 1.335093106856],
            [1291093200000, 1.3302565104985],
            [1293771600000, 1.340824670897],
            [1296450000000, 0],
            [1298869200000, 0],
            [1301544000000, 0],
            [1304136000000, 0],
            [1306814400000, 0],
            [1309406400000, 0],
            [1312084800000, 0],
            [1314763200000, 0],
            [1317355200000, 4.4583692315],
            [1320033600000, 3.6493043348059],
            [1322629200000, 3.8610064091761],
            [1325307600000, 5.5144800685202],
            [1327986000000, 5.1750695220792],
            [1330491600000, 5.6710066952691],
            [1333166400000, 8.5658461590953],
            [1335758400000, 8.6135447714243],
            [1338436800000, 8.0231460925212]
          ]
        }, {
          "key": "Energy",
          "values": [
            [1138683600000, 1.544303464167],
            [1141102800000, 1.4387289432421],
            [1143781200000, 0],
            [1146369600000, 0],
            [1149048000000, 0],
            [1151640000000, 1.328626801128],
            [1154318400000, 1.2874050802627],
            [1156996800000, 1.0872743105593],
            [1159588800000, 0.96042562635813],
            [1162270800000, 0.93139372870616],
            [1164862800000, 0.94432167305385],
            [1167541200000, 1.277750166208],
            [1170219600000, 1.2204893886811],
            [1172638800000, 1.207489123122],
            [1175313600000, 1.2490651414113],
            [1177905600000, 1.2593129913052],
            [1180584000000, 1.373329808388],
```

```
            [1183176000000, 0],
            [1185854400000, 0],
            [1188532800000, 0],
            [1191124800000, 0],
            [1193803200000, 0],
            [1196398800000, 0],
            [1199077200000, 0],
            [1201755600000, 0],
            [1204261200000, 0],
            [1206936000000, 0],
            [1209528000000, 0],
            [1212206400000, 0],
            [1214798400000, 0],
            [1217476800000, 0],
            [1220155200000, 0],
            [1222747200000, 1.4516108933695],
            [1225425600000, 1.1856025268225],
            [1228021200000, 1.3430470355439],
            [1230699600000, 2.2752595354509],
            [1233378000000, 2.4031560010523],
            [1235797200000, 2.0822430731926],
            [1238472000000, 1.5640902826938],
            [1241064000000, 1.5812873972356],
            [1243742400000, 1.9462448548894],
            [1246334400000, 2.9464870223957],
            [1249012800000, 3.0744699383222],
            [1251691200000, 2.9422304628446],
            [1254283200000, 2.7503075599999],
            [1256961600000, 2.6506701800427],
            [1259557200000, 2.8005425319977],
            [1262235600000, 2.6816184971185],
            [1264914000000, 2.681206271327],
            [1267333200000, 2.8195488011259],
            [1270008000000, 0],
            [1272600000000, 0],
            [1275278400000, 0],
            [1277870400000, 1.0687057346382],
            [1280548800000, 1.2539400544134],
            [1283227200000, 1.1862969445955],
            [1285819200000, 0],
            [1288497600000, 0],
            [1291093200000, 0],
            [1293771600000, 0],
            [1296450000000, 1.941972859484],
            [1298869200000, 2.1142247697552],
            [1301544000000, 2.3788590206824],
            [1304136000000, 2.5337302877545],
            [1306814400000, 2.3163370395199],
            [1309406400000, 2.0645451843195],
            [1312084800000, 2.1004446672411],
            [1314763200000, 3.6301875804303],
            [1317355200000, 2.454204664652],
            [1320033600000, 2.196082370894],
            [1322629200000, 2.3358418255202],
            [1325307600000, 0],
            [1327986000000, 0],
            [1330491600000, 0],
            [1333166400000, 0.39001201038526],
```

```
            [1335758400000, 0.30945472725559],
            [1338436800000, 0.31062439305591]
          ]
        }, {
          "key": "Financials",
          "values": [
            [1138683600000, 13.356778764352],
            [1141102800000, 13.611196863271],
            [1143781200000, 6.895903006119],
            [1146369600000, 6.9939633271352],
            [1149048000000, 6.7241510257675],
            [1151640000000, 5.5611293669516],
            [1154318400000, 5.6086488714041],
            [1156996800000, 5.4962849907033],
            [1159588800000, 6.9193153169279],
            [1162270800000, 7.0016334389777],
            [1164862800000, 6.7865422443273],
            [1167541200000, 9.0006454225383],
            [1170219600000, 9.2233916171431],
            [1172638800000, 8.8929316009479],
            [1175313600000, 10.345937520404],
            [1177905600000, 10.075914677026],
            [1180584000000, 10.089006188111],
            [1183176000000, 10.598330295008],
            [1185854400000, 9.968954653301],
            [1188532800000, 9.7740580198146],
            [1191124800000, 10.558483060626],
            [1193803200000, 9.9314651823603],
            [1196398800000, 9.3997715873769],
            [1199077200000, 8.4086493387262],
            [1201755600000, 8.9698309085926],
            [1204261200000, 8.2778357995396],
            [1206936000000, 8.8585045600123],
            [1209528000000, 8.7013756413322],
            [1212206400000, 7.7933605469443],
            [1214798400000, 7.0236183483064],
            [1217476800000, 6.9873088186829],
            [1220155200000, 6.8031713070097],
            [1222747200000, 6.6869531315723],
            [1225425600000, 6.138256993963],
            [1228021200000, 5.6434994016354],
            [1230699600000, 5.495220262512],
            [1233378000000, 4.6885326869846],
            [1235797200000, 4.4524349883438],
            [1238472000000, 5.6766520778185],
            [1241064000000, 5.7675774480752],
            [1243742400000, 5.7882863168337],
            [1246334400000, 7.2666010034924],
            [1249012800000, 7.519182132226],
            [1251691200000, 7.849651451445],
            [1254283200000, 10.383992037985],
            [1256961600000, 9.0653691861818],
            [1259557200000, 9.6705248324159],
            [1262235600000, 10.856380561349],
            [1264914000000, 11.27452370892],
            [1267333200000, 11.754156529088],
            [1270008000000, 8.2870811422456],
            [1272600000000, 8.0210264360699],
```

```
            [1275278400000, 7.5375074474865],
            [1277870400000, 8.3419527338039],
            [1280548800000, 9.4197471818443],
            [1283227200000, 8.7321733185797],
            [1285819200000, 9.6627062648126],
            [1288497600000, 10.187962234549],
            [1291093200000, 9.8144201733476],
            [1293771600000, 10.275723361713],
            [1296450000000, 16.796066079353],
            [1298869200000, 17.543254984075],
            [1301544000000, 16.673660675084],
            [1304136000000, 17.963944353609],
            [1306814400000, 16.637740867211],
            [1309406400000, 15.84857094609],
            [1312084800000, 14.767303362182],
            [1314763200000, 24.778452182432],
            [1317355200000, 18.370353229999],
            [1320033600000, 15.2531374291],
            [1322629200000, 14.989600840649],
            [1325307600000, 16.052539160125],
            [1327986000000, 16.424390322793],
            [1330491600000, 17.884020741105],
            [1333166400000, 7.1424929577921],
            [1335758400000, 7.8076213051482],
            [1338436800000, 7.2462684949232]
          ]
        }, {
          "key": "Health Care",
          "values": [
            [1138683600000, 14.212410956029],
            [1141102800000, 13.973193618249],
            [1143781200000, 15.218233920665],
            [1146369600000, 14.38210972745],
            [1149048000000, 13.894310878491],
            [1151640000000, 15.593086090032],
            [1154318400000, 16.244839695188],
            [1156996800000, 16.017088850646],
            [1159588800000, 14.183951830055],
            [1162270800000, 14.148523245697],
            [1164862800000, 13.424326059972],
            [1167541200000, 12.974450435753],
            [1170219600000, 13.23247041802],
            [1172638800000, 13.318762655574],
            [1175313600000, 15.961407746104],
            [1177905600000, 16.287714639805],
            [1180584000000, 16.246590583889],
            [1183176000000, 17.564505594809],
            [1185854400000, 17.872725373165],
            [1188532800000, 18.018998508757],
            [1191124800000, 15.584518016603],
            [1193803200000, 15.480850647181],
            [1196398800000, 15.699120036984],
            [1199077200000, 19.184281817226],
            [1201755600000, 19.691226605207],
            [1204261200000, 18.982314051295],
            [1206936000000, 18.707820309008],
            [1209528000000, 17.459630929761],
            [1212206400000, 16.500616076782],
```

```
            [1214798400000, 18.086324003979],
            [1217476800000, 18.929464156258],
            [1220155200000, 18.233728682084],
            [1222747200000, 16.315776297325],
            [1225425600000, 14.63289219025],
            [1228021200000, 14.667835024478],
            [1230699600000, 13.946993947308],
            [1233378000000, 14.394304684397],
            [1235797200000, 13.724462792967],
            [1238472000000, 10.930879035806],
            [1241064000000, 9.8339915513708],
            [1243742400000, 10.053858541872],
            [1246334400000, 11.786998438287],
            [1249012800000, 11.780994901769],
            [1251691200000, 11.305889670276],
            [1254283200000, 10.918452290083],
            [1256961600000, 9.6811395055706],
            [1259557200000, 10.971529744038],
            [1262235600000, 13.330210480209],
            [1264914000000, 14.592637568961],
            [1267333200000, 14.605329141157],
            [1270008000000, 13.936853794037],
            [1272600000000, 12.189480759072],
            [1275278400000, 11.676151385046],
            [1277870400000, 13.058852800017],
            [1280548800000, 13.62891543203],
            [1283227200000, 13.811107569918],
            [1285819200000, 13.786494560787],
            [1288497600000, 14.04516285753],
            [1291093200000, 13.697412447288],
            [1293771600000, 13.677681376221],
            [1296450000000, 19.961511864531],
            [1298869200000, 21.049198298158],
            [1301544000000, 22.687631094008],
            [1304136000000, 25.469010617433],
            [1306814400000, 24.883799437121],
            [1309406400000, 24.203843814248],
            [1312084800000, 22.138760964038],
            [1314763200000, 16.034636966228],
            [1317355200000, 15.394958944556],
            [1320033600000, 12.625642461969],
            [1322629200000, 12.973735699739],
            [1325307600000, 15.786018336149],
            [1327986000000, 15.227368020134],
            [1330491600000, 15.899752650734],
            [1333166400000, 18.994731295388],
            [1335758400000, 18.450055817702],
            [1338436800000, 17.863719889669]
          ]
        }, {
          "key": "Industrials",
          "values": [
            [1138683600000, 7.1590087090398],
            [1141102800000, 7.1297210970108],
            [1143781200000, 5.5774588290586],
            [1146369600000, 5.4977254491156],
            [1149048000000, 5.5138153113634],
            [1151640000000, 4.3198084032122],
```

```
            [1154318400000, 3.9179295839125],
            [1156996800000, 3.8110093051479],
            [1159588800000, 5.5629020916939],
            [1162270800000, 5.7241673711336],
            [1164862800000, 5.4715049695004],
            [1167541200000, 4.9193763571618],
            [1170219600000, 5.136053947247],
            [1172638800000, 5.1327258759766],
            [1175313600000, 5.1888943925082],
            [1177905600000, 5.5191481293345],
            [1180584000000, 5.6093625614921],
            [1183176000000, 4.2706312987397],
            [1185854400000, 4.4453235132117],
            [1188532800000, 4.6228003109761],
            [1191124800000, 5.0645764756954],
            [1193803200000, 5.0723447230959],
            [1196398800000, 5.1457765818846],
            [1199077200000, 5.4067851597282],
            [1201755600000, 5.472241916816],
            [1204261200000, 5.3742740389688],
            [1206936000000, 6.251751933664],
            [1209528000000, 6.1406852153472],
            [1212206400000, 5.8164385627465],
            [1214798400000, 5.4255846656171],
            [1217476800000, 5.3738499417204],
            [1220155200000, 5.1815627753979],
            [1222747200000, 5.0305983235349],
            [1225425600000, 4.6823058607165],
            [1228021200000, 4.5941481589093],
            [1230699600000, 5.4669598474575],
            [1233378000000, 5.1249037357],
            [1235797200000, 4.3504421250742],
            [1238472000000, 4.6260881026002],
            [1241064000000, 5.0140402458946],
            [1243742400000, 4.7458462454774],
            [1246334400000, 6.0437019654564],
            [1249012800000, 6.4595216249754],
            [1251691200000, 6.6420468254155],
            [1254283200000, 5.8927271960913],
            [1256961600000, 5.4712108838003],
            [1259557200000, 6.1220254207747],
            [1262235600000, 5.5385935169255],
            [1264914000000, 5.7383377612639],
            [1267333200000, 6.1715976730415],
            [1270008000000, 4.0102262681174],
            [1272600000000, 3.769389679692],
            [1275278400000, 3.5301571031152],
            [1277870400000, 2.7660252652526],
            [1280548800000, 3.1409983385775],
            [1283227200000, 3.0528024863055],
            [1285819200000, 4.3126123157971],
            [1288497600000, 4.594654041683],
            [1291093200000, 4.5424126126793],
            [1293771600000, 4.7790043987302],
            [1296450000000, 7.4969154058289],
            [1298869200000, 7.9424751557821],
            [1301544000000, 7.1560736250547],
            [1304136000000, 7.9478117337855],
```

```
            [1306814400000, 7.4109214848895],
            [1309406400000, 7.5966457641101],
            [1312084800000, 7.165754444071],
            [1314763200000, 5.4816702524302],
            [1317355200000, 4.9893656089584],
            [1320033600000, 4.498385105327],
            [1322629200000, 4.6776090358151],
            [1325307600000, 8.1350814368063],
            [1327986000000, 8.0732769990652],
            [1330491600000, 8.5602340387277],
            [1333166400000, 5.1293714074325],
            [1335758400000, 5.2586794619016],
            [1338436800000, 5.1100853569977]
          ]
        }, {
          "key": "Information Technology",
          "values": [
            [1138683600000, 13.242301508051],
            [1141102800000, 12.863536342042],
            [1143781200000, 21.034044171629],
            [1146369600000, 21.419084618803],
            [1149048000000, 21.142678863691],
            [1151640000000, 26.568489677529],
            [1154318400000, 24.839144939905],
            [1156996800000, 25.456187462167],
            [1159588800000, 26.350164502826],
            [1162270800000, 26.47833320519],
            [1164862800000, 26.425979547847],
            [1167541200000, 28.191461582256],
            [1170219600000, 28.930307448808],
            [1172638800000, 29.521413891117],
            [1175313600000, 28.188285966466],
            [1177905600000, 27.704619625832],
            [1180584000000, 27.490862424829],
            [1183176000000, 28.770679721286],
            [1185854400000, 29.060480671449],
            [1188532800000, 28.240998844973],
            [1191124800000, 33.004893194127],
            [1193803200000, 34.075180359928],
            [1196398800000, 32.548560664833],
            [1199077200000, 30.629727432728],
            [1201755600000, 28.642858788159],
            [1204261200000, 27.973575227842],
            [1206936000000, 27.393351882726],
            [1209528000000, 28.476095288523],
            [1212206400000, 29.29667866426],
            [1214798400000, 29.222333802896],
            [1217476800000, 28.092966093843],
            [1220155200000, 28.107159262922],
            [1222747200000, 25.482974832098],
            [1225425600000, 21.208115993834],
            [1228021200000, 20.295043095268],
            [1230699600000, 15.925754618401],
            [1233378000000, 17.162864628346],
            [1235797200000, 17.084345773174],
            [1238472000000, 22.246007102281],
            [1241064000000, 24.530543998509],
            [1243742400000, 25.084184918242],
```

```
            [1246334400000, 16.606166527358],
            [1249012800000, 17.239620011628],
            [1251691200000, 17.336739127379],
            [1254283200000, 25.478492475753],
            [1256961600000, 23.017152085245],
            [1259557200000, 25.617745423683],
            [1262235600000, 24.061133998642],
            [1264914000000, 23.223933318644],
            [1267333200000, 24.425887263937],
            [1270008000000, 35.501471156693],
            [1272600000000, 33.775013878676],
            [1275278400000, 30.417993630285],
            [1277870400000, 30.023598978467],
            [1280548800000, 33.327519522436],
            [1283227200000, 31.963388450371],
            [1285819200000, 30.498967232092],
            [1288497600000, 32.403696817912],
            [1291093200000, 31.47736071922],
            [1293771600000, 31.53259666241],
            [1296450000000, 41.760282761548],
            [1298869200000, 45.605771243237],
            [1301544000000, 39.986557966215],
            [1304136000000, 43.846330510051],
            [1306814400000, 39.857316881857],
            [1309406400000, 37.675127768208],
            [1312084800000, 35.775077970313],
            [1314763200000, 48.631009702577],
            [1317355200000, 42.830831754505],
            [1320033600000, 35.611502589362],
            [1322629200000, 35.320136981738],
            [1325307600000, 31.564136901516],
            [1327986000000, 32.074407502433],
            [1330491600000, 35.053013769976],
            [1333166400000, 26.434568573937],
            [1335758400000, 25.305617871002],
            [1338436800000, 24.520919418236]
          ]
        }, {
          "key": "Materials",
          "values": [
            [1138683600000, 5.5806167415681],
            [1141102800000, 5.4539047069985],
            [1143781200000, 7.6728842432362],
            [1146369600000, 7.719946716654],
            [1149048000000, 8.0144619912942],
            [1151640000000, 7.942223133434],
            [1154318400000, 8.3998279827444],
            [1156996800000, 8.532324572605],
            [1159588800000, 4.7324285199763],
            [1162270800000, 4.7402397487697],
            [1164862800000, 4.9042069355168],
            [1167541200000, 5.9583963430882],
            [1170219600000, 6.3693899239171],
            [1172638800000, 6.261153903813],
            [1175313600000, 5.3443942184584],
            [1177905600000, 5.4932111235361],
            [1180584000000, 5.5747393101109],
            [1183176000000, 5.3833633060013],
```

```
            [1185854400000, 5.5125898831832],
            [1188532800000, 5.8116112661327],
            [1191124800000, 4.3962296939996],
            [1193803200000, 4.6967663605521],
            [1196398800000, 4.7963004350914],
            [1199077200000, 4.1817985183351],
            [1201755600000, 4.3797643870182],
            [1204261200000, 4.6966642197965],
            [1206936000000, 4.3609995132565],
            [1209528000000, 4.4736290996496],
            [1212206400000, 4.3749762738128],
            [1214798400000, 3.3274661194507],
            [1217476800000, 3.0316184691337],
            [1220155200000, 2.5718140204728],
            [1222747200000, 2.7034994044603],
            [1225425600000, 2.2033786591364],
            [1228021200000, 1.9850621240805],
            [1230699600000, 0],
            [1233378000000, 0],
            [1235797200000, 0],
            [1238472000000, 0],
            [1241064000000, 0],
            [1243742400000, 0],
            [1246334400000, 0],
            [1249012800000, 0],
            [1251691200000, 0],
            [1254283200000, 0.44495950017788],
            [1256961600000, 0.33945469262483],
            [1259557200000, 0.38348269455195],
            [1262235600000, 0],
            [1264914000000, 0],
            [1267333200000, 0],
            [1270008000000, 0],
            [1272600000000, 0],
            [1275278400000, 0],
            [1277870400000, 0],
            [1280548800000, 0],
            [1283227200000, 0],
            [1285819200000, 0],
            [1288497600000, 0],
            [1291093200000, 0],
            [1293771600000, 0],
            [1296450000000, 0.52216435716176],
            [1298869200000, 0.59275786698454],
            [1301544000000, 0],
            [1304136000000, 0],
            [1306814400000, 0],
            [1309406400000, 0],
            [1312084800000, 0],
            [1314763200000, 0],
            [1317355200000, 0],
            [1320033600000, 0],
            [1322629200000, 0],
            [1325307600000, 0],
            [1327986000000, 0],
            [1330491600000, 0],
            [1333166400000, 0],
            [1335758400000, 0],
```

```
            [1338436800000, 0]
          ]
        }, {
          "key": "Telecommunication Services",
          "values": [
            [1138683600000, 3.7056975170243],
            [1141102800000, 3.7561118692318],
            [1143781200000, 2.861913700854],
            [1146369600000, 2.9933744103381],
            [1149048000000, 2.7127537218463],
            [1151640000000, 3.1195497076283],
            [1154318400000, 3.4066964004508],
            [1156996800000, 3.3754571113569],
            [1159588800000, 2.2965579982924],
            [1162270800000, 2.4486818633018],
            [1164862800000, 2.4002308848517],
            [1167541200000, 1.9649579750349],
            [1170219600000, 1.9385263638056],
            [1172638800000, 1.9128975336387],
            [1175313600000, 2.3412869836298],
            [1177905600000, 2.4337870351445],
            [1180584000000, 2.62179703171],
            [1183176000000, 3.2642864957929],
            [1185854400000, 3.3200396223709],
            [1188532800000, 3.3934212707572],
            [1191124800000, 4.2822327088179],
            [1193803200000, 4.1474964228541],
            [1196398800000, 4.1477082879801],
            [1199077200000, 5.2947122916128],
            [1201755600000, 5.2919843508028],
            [1204261200000, 5.1989783050309],
            [1206936000000, 3.5603057673513],
            [1209528000000, 3.3009087690692],
            [1212206400000, 3.1784852603792],
            [1214798400000, 4.5889503538868],
            [1217476800000, 4.401779617494],
            [1220155200000, 4.2208301828278],
            [1222747200000, 3.89396671475],
            [1225425600000, 3.0423832241354],
            [1228021200000, 3.135520611578],
            [1230699600000, 1.9631418164089],
            [1233378000000, 1.8963543874958],
            [1235797200000, 1.8266636017025],
            [1238472000000, 0.93136635895188],
            [1241064000000, 0.92737801918888],
            [1243742400000, 0.97591889805002],
            [1246334400000, 2.6841193805515],
            [1249012800000, 2.5664341140531],
            [1251691200000, 2.3887523699873],
            [1254283200000, 1.1737801663681],
            [1256961600000, 1.0953582317281],
            [1259557200000, 1.2495674976653],
            [1262235600000, 0.36607452464754],
            [1264914000000, 0.3548719047291],
            [1267333200000, 0.36769242398939],
            [1270008000000, 0],
            [1272600000000, 0],
            [1275278400000, 0],
```

```
            [1277870400000, 0],
            [1280548800000, 0],
            [1283227200000, 0],
            [1285819200000, 0.85450741275337],
            [1288497600000, 0.91360317921637],
            [1291093200000, 0.89647678692269],
            [1293771600000, 0.87800687192639],
            [1296450000000, 0],
            [1298869200000, 0],
            [1301544000000, 0.43668720882994],
            [1304136000000, 0.4756523602692],
            [1306814400000, 0.46947368328469],
            [1309406400000, 0.45138896152316],
            [1312084800000, 0.43828726648117],
            [1314763200000, 2.0820861395316],
            [1317355200000, 0.9364411075395],
            [1320033600000, 0.60583907839773],
            [1322629200000, 0.61096950747437],
            [1325307600000, 0],
            [1327986000000, 0],
            [1330491600000, 0],
            [1333166400000, 0],
            [1335758400000, 0],
            [1338436800000, 0]
          ]
        }, {
          "key": "Utilities",
          "values": [
            [1138683600000, 0],
            [1141102800000, 0],
            [1143781200000, 0],
            [1146369600000, 0],
            [1149048000000, 0],
            [1151640000000, 0],
            [1154318400000, 0],
            [1156996800000, 0],
            [1159588800000, 0],
            [1162270800000, 0],
            [1164862800000, 0],
            [1167541200000, 0],
            [1170219600000, 0],
            [1172638800000, 0],
            [1175313600000, 0],
            [1177905600000, 0],
            [1180584000000, 0],
            [1183176000000, 0],
            [1185854400000, 0],
            [1188532800000, 0],
            [1191124800000, 0],
            [1193803200000, 0],
            [1196398800000, 0],
            [1199077200000, 0],
            [1201755600000, 0],
            [1204261200000, 0],
            [1206936000000, 0],
            [1209528000000, 0],
            [1212206400000, 0],
            [1214798400000, 0],
```

```
            [1217476800000, 0],
            [1220155200000, 0],
            [1222747200000, 0],
            [1225425600000, 0],
            [1228021200000, 0],
            [1230699600000, 0],
            [1233378000000, 0],
            [1235797200000, 0],
            [1238472000000, 0],
            [1241064000000, 0],
            [1243742400000, 0],
            [1246334400000, 0],
            [1249012800000, 0],
            [1251691200000, 0],
            [1254283200000, 0],
            [1256961600000, 0],
            [1259557200000, 0],
            [1262235600000, 0],
            [1264914000000, 0],
            [1267333200000, 0],
            [1270008000000, 0],
            [1272600000000, 0],
            [1275278400000, 0],
            [1277870400000, 0],
            [1280548800000, 0],
            [1283227200000, 0],
            [1285819200000, 0],
            [1288497600000, 0],
            [1291093200000, 0],
            [1293771600000, 0],
            [1296450000000, 0],
            [1298869200000, 0],
            [1301544000000, 0],
            [1304136000000, 0],
            [1306814400000, 0],
            [1309406400000, 0],
            [1312084800000, 0],
            [1314763200000, 0],
            [1317355200000, 0],
            [1320033600000, 0],
            [1322629200000, 0],
            [1325307600000, 0],
            [1327986000000, 0],
            [1330491600000, 0],
            [1333166400000, 0],
            [1335758400000, 0],
            [1338436800000, 0]
          ]
        }];
      }
    },

    chart9config: {
      type: Object,
      value: function() {
        return {
          useInteractiveGuideline: true,
          x: function(d) { return d[0] },
```

```
          y: function(d) { return d[1] },
          duration: 300,
          controlLabels: {stacked: "Stacked"}
        }
      }
    },

    chart9JSConfig: {
      type: Function,
      value: function() {
        return function() {
          this._chart.xAxis
            .tickFormat(function(d) { return d3.time.format('%x')(new Date(d)) });
          this._chart.yAxis
            .tickFormat(d3.format(',.4f'));
          this._chart.legend.vers('furious');
        }
      }
    }
  }
});
```

The result will look like this:

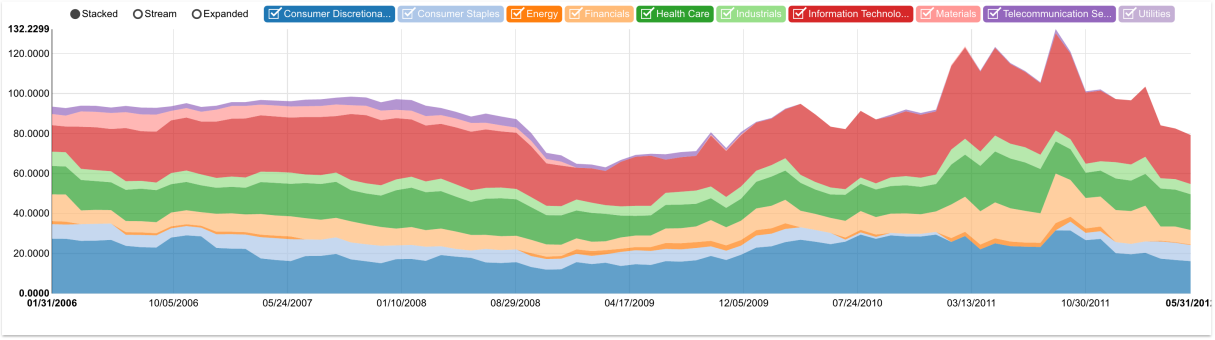


Chart9: A Candlestick charts

HTML & Javascript

Edit src/html/elements/chart-examples/chart-examples.html

/src/html/elements/chart-examples/chart-examples.html

```
<dom-module id="chart-examples">
  <template>

    <div horizontal layout flex>
      <paper-material elevation="1" flex horizontal layout>
        <candlestick-chart flex data="{{chart11data}}" config="{{chart11config}}" configure-chart="{{chart11JSConfig}}"></candlestick-chart>
      </paper-material>
    </div>

  </template>
</dom-module>
```

Now edit /src/html/elements/chart-examples/javascript/chart-examples.js

/src/html/elements/chart-examples/javascript/chart-examples.js

```
Polymer({
  properties: {
    chart11data: {
      type: Array,
      value: function() {
        return [{
          values: [{
            "date": 15854,
            "open": 165.42,
            "high": 165.8,
            "low": 164.34,
            "close": 165.22,
            "volume": 160363400,
            "adjusted": 164.35
          }, {
            "date": 15855,
            "open": 165.35,
            "high": 166.59,
            "low": 165.22,
            "close": 165.83,
            "volume": 107793800,
            "adjusted": 164.96
          }, {
            "date": 15856,
            "open": 165.37,
            "high": 166.31,
            "low": 163.13,
            "close": 163.45,
            "volume": 176850100,
            "adjusted": 162.59
          }, {
            "date": 15859,
            "open": 163.83,
            "high": 164.46,
            "low": 162.66,
            "close": 164.35,
            "volume": 168390700,
            "adjusted": 163.48
          }, {
            "date": 15860,
            "open": 164.44,
            "high": 165.1,
            "low": 162.73,
            "close": 163.56,
            "volume": 157631500,
            "adjusted": 162.7
          }, {
            "date": 15861,
            "open": 163.09,
            "high": 163.42,
            "low": 161.13,
            "close": 161.27,
            "volume": 211737800,
            "adjusted": 160.42
          }, {
```

```
            "date": 15862,
            "open": 161.2,
            "high": 162.74,
            "low": 160.25,
            "close": 162.73,
            "volume": 200225500,
            "adjusted": 161.87
          }, {
            "date": 15863,
            "open": 163.85,
            "high": 164.95,
            "low": 163.14,
            "close": 164.8,
            "volume": 188337800,
            "adjusted": 163.93
          }, {
            "date": 15866,
            "open": 165.31,
            "high": 165.4,
            "low": 164.37,
            "close": 164.8,
            "volume": 105667100,
            "adjusted": 163.93
          }, {
            "date": 15867,
            "open": 163.3,
            "high": 164.54,
            "low": 162.74,
            "close": 163.1,
            "volume": 159505400,
            "adjusted": 162.24
          }, {
            "date": 15868,
            "open": 164.22,
            "high": 164.39,
            "low": 161.6,
            "close": 161.75,
            "volume": 177361500,
            "adjusted": 160.9
          }, {
            "date": 15869,
            "open": 161.66,
            "high": 164.5,
            "low": 161.3,
            "close": 164.21,
            "volume": 163587800,
            "adjusted": 163.35
          }, {
            "date": 15870,
            "open": 164.03,
            "high": 164.67,
            "low": 162.91,
            "close": 163.18,
            "volume": 141197500,
            "adjusted": 162.32
          }, {
            "date": 15873,
            "open": 164.29,
```

```
            "high": 165.22,
            "low": 163.22,
            "close": 164.44,
            "volume": 136295600,
            "adjusted": 163.57
          }, {
            "date": 15874,
            "open": 164.53,
            "high": 165.99,
            "low": 164.52,
            "close": 165.74,
            "volume": 114695600,
            "adjusted": 164.87
          }, {
            "date": 15875,
            "open": 165.6,
            "high": 165.89,
            "low": 163.38,
            "close": 163.45,
            "volume": 206149500,
            "adjusted": 162.59
          }, {
            "date": 15876,
            "open": 161.86,
            "high": 163.47,
            "low": 158.98,
            "close": 159.4,
            "volume": 321255900,
            "adjusted": 158.56
          }, {
            "date": 15877,
            "open": 159.64,
            "high": 159.76,
            "low": 157.47,
            "close": 159.07,
            "volume": 271956800,
            "adjusted": 159.07
          }, {
            "date": 15880,
            "open": 157.41,
            "high": 158.43,
            "low": 155.73,
            "close": 157.06,
            "volume": 222329000,
            "adjusted": 157.06
          }, {
            "date": 15881,
            "open": 158.48,
            "high": 160.1,
            "low": 157.42,
            "close": 158.57,
            "volume": 162262200,
            "adjusted": 158.57
          }, {
            "date": 15882,
            "open": 159.87,
            "high": 160.5,
            "low": 159.25,
```

```
            "close": 160.14,
            "volume": 134848000,
            "adjusted": 160.14
          }, {
            "date": 15883,
            "open": 161.1,
            "high": 161.82,
            "low": 160.95,
            "close": 161.08,
            "volume": 129483700,
            "adjusted": 161.08
          }, {
            "date": 15884,
            "open": 160.63,
            "high": 161.4,
            "low": 159.86,
            "close": 160.42,
            "volume": 160402900,
            "adjusted": 160.42
          }, {
            "date": 15887,
            "open": 161.26,
            "high": 162.48,
            "low": 161.08,
            "close": 161.36,
            "volume": 131954800,
            "adjusted": 161.36
          }, {
            "date": 15888,
            "open": 161.12,
            "high": 162.3,
            "low": 160.5,
            "close": 161.21,
            "volume": 154863700,
            "adjusted": 161.21
          }, {
            "date": 15889,
            "open": 160.48,
            "high": 161.77,
            "low": 160.22,
            "close": 161.28,
            "volume": 75216400,
            "adjusted": 161.28
          }, {
            "date": 15891,
            "open": 162.47,
            "high": 163.08,
            "low": 161.3,
            "close": 163.02,
            "volume": 122416900,
            "adjusted": 163.02
          }, {
            "date": 15894,
            "open": 163.86,
            "high": 164.39,
            "low": 163.08,
            "close": 163.95,
            "volume": 108092500,
```

```
            "adjusted": 163.95
          }, {
            "date": 15895,
            "open": 164.98,
            "high": 165.33,
            "low": 164.27,
            "close": 165.13,
            "volume": 119298000,
            "adjusted": 165.13
          }, {
            "date": 15896,
            "open": 164.97,
            "high": 165.75,
            "low": 164.63,
            "close": 165.19,
            "volume": 121410100,
            "adjusted": 165.19
          }, {
            "date": 15897,
            "open": 167.11,
            "high": 167.61,
            "low": 165.18,
            "close": 167.44,
            "volume": 135592200,
            "adjusted": 167.44
          }, {
            "date": 15898,
            "open": 167.39,
            "high": 167.93,
            "low": 167.13,
            "close": 167.51,
            "volume": 104212700,
            "adjusted": 167.51
          }, {
            "date": 15901,
            "open": 167.97,
            "high": 168.39,
            "low": 167.68,
            "close": 168.15,
            "volume": 69450600,
            "adjusted": 168.15
          }, {
            "date": 15902,
            "open": 168.26,
            "high": 168.36,
            "low": 167.07,
            "close": 167.52,
            "volume": 88702100,
            "adjusted": 167.52
          }, {
            "date": 15903,
            "open": 168.16,
            "high": 168.48,
            "low": 167.73,
            "close": 167.95,
            "volume": 92873900,
            "adjusted": 167.95
          }, {
```

```
            "date": 15904,
            "open": 168.31,
            "high": 169.27,
            "low": 168.2,
            "close": 168.87,
            "volume": 103620100,
            "adjusted": 168.87
          }, {
            "date": 15905,
            "open": 168.52,
            "high": 169.23,
            "low": 168.31,
            "close": 169.17,
            "volume": 103831700,
            "adjusted": 169.17
          }, {
            "date": 15908,
            "open": 169.41,
            "high": 169.74,
            "low": 169.01,
            "close": 169.5,
            "volume": 79428600,
            "adjusted": 169.5
          }, {
            "date": 15909,
            "open": 169.8,
            "high": 169.83,
            "low": 169.05,
            "close": 169.14,
            "volume": 80829700,
            "adjusted": 169.14
          }, {
            "date": 15910,
            "open": 169.79,
            "high": 169.86,
            "low": 168.18,
            "close": 168.52,
            "volume": 112914000,
            "adjusted": 168.52
          }, {
            "date": 15911,
            "open": 168.22,
            "high": 169.08,
            "low": 167.94,
            "close": 168.93,
            "volume": 111088600,
            "adjusted": 168.93
          }, {
            "date": 15912,
            "open": 168.22,
            "high": 169.16,
            "low": 167.52,
            "close": 169.11,
            "volume": 107814600,
            "adjusted": 169.11
          }, {
            "date": 15915,
            "open": 168.68,
```

```
            "high": 169.06,
            "low": 168.11,
            "close": 168.59,
            "volume": 79695000,
            "adjusted": 168.59
          }, {
            "date": 15916,
            "open": 169.1,
            "high": 169.28,
            "low": 168.19,
            "close": 168.59,
            "volume": 85209600,
            "adjusted": 168.59
          }, {
            "date": 15917,
            "open": 168.94,
            "high": 169.85,
            "low": 168.49,
            "close": 168.71,
            "volume": 142388700,
            "adjusted": 168.71
          }, {
            "date": 15918,
            "open": 169.99,
            "high": 170.81,
            "low": 169.9,
            "close": 170.66,
            "volume": 110438400,
            "adjusted": 170.66
          }, {
            "date": 15919,
            "open": 170.28,
            "high": 170.97,
            "low": 170.05,
            "close": 170.95,
            "volume": 91116700,
            "adjusted": 170.95
          }, {
            "date": 15922,
            "open": 170.57,
            "high": 170.96,
            "low": 170.35,
            "close": 170.7,
            "volume": 54072700,
            "adjusted": 170.7
          }, {
            "date": 15923,
            "open": 170.37,
            "high": 170.74,
            "low": 169.35,
            "close": 169.73,
            "volume": 87495000,
            "adjusted": 169.73
          }, {
            "date": 15924,
            "open": 169.19,
            "high": 169.43,
            "low": 168.55,
```

```
            "close": 169.18,
            "volume": 84854700,
            "adjusted": 169.18
          }, {
            "date": 15925,
            "open": 169.98,
            "high": 170.18,
            "low": 168.93,
            "close": 169.8,
            "volume": 102181300,
            "adjusted": 169.8
          }, {
            "date": 15926,
            "open": 169.58,
            "high": 170.1,
            "low": 168.72,
            "close": 169.31,
            "volume": 91757700,
            "adjusted": 169.31
          }, {
            "date": 15929,
            "open": 168.46,
            "high": 169.31,
            "low": 168.38,
            "close": 169.11,
            "volume": 68593300,
            "adjusted": 169.11
          }, {
            "date": 15930,
            "open": 169.41,
            "high": 169.9,
            "low": 168.41,
            "close": 169.61,
            "volume": 80806000,
            "adjusted": 169.61
          }, {
            "date": 15931,
            "open": 169.53,
            "high": 169.8,
            "low": 168.7,
            "close": 168.74,
            "volume": 79829200,
            "adjusted": 168.74
          }, {
            "date": 15932,
            "open": 167.41,
            "high": 167.43,
            "low": 166.09,
            "close": 166.38,
            "volume": 152931800,
            "adjusted": 166.38
          }, {
            "date": 15933,
            "open": 166.06,
            "high": 166.63,
            "low": 165.5,
            "close": 165.83,
            "volume": 130868200,
```

```
            "adjusted": 165.83
          }, {
            "date": 15936,
            "open": 165.64,
            "high": 166.21,
            "low": 164.76,
            "close": 164.77,
            "volume": 96437600,
            "adjusted": 164.77
          }, {
            "date": 15937,
            "open": 165.04,
            "high": 166.2,
            "low": 164.86,
            "close": 165.58,
            "volume": 89294400,
            "adjusted": 165.58
          }, {
            "date": 15938,
            "open": 165.12,
            "high": 166.03,
            "low": 164.19,
            "close": 164.56,
            "volume": 159530500,
            "adjusted": 164.56
          }, {
            "date": 15939,
            "open": 164.9,
            "high": 166.3,
            "low": 164.89,
            "close": 166.06,
            "volume": 101471400,
            "adjusted": 166.06
          }, {
            "date": 15940,
            "open": 166.55,
            "high": 166.83,
            "low": 165.77,
            "close": 166.62,
            "volume": 90888900,
            "adjusted": 166.62
          }, {
            "date": 15943,
            "open": 166.79,
            "high": 167.3,
            "low": 165.89,
            "close": 166,
            "volume": 89702100,
            "adjusted": 166
          }, {
            "date": 15944,
            "open": 164.36,
            "high": 166,
            "low": 163.21,
            "close": 163.33,
            "volume": 158619400,
            "adjusted": 163.33
          }, {
```

```
            "date": 15945,
            "open": 163.26,
            "high": 164.49,
            "low": 163.05,
            "close": 163.91,
            "volume": 108113000,
            "adjusted": 163.91
          }, {
            "date": 15946,
            "open": 163.55,
            "high": 165.04,
            "low": 163.4,
            "close": 164.17,
            "volume": 119200500,
            "adjusted": 164.17
          }, {
            "date": 15947,
            "open": 164.51,
            "high": 164.53,
            "low": 163.17,
            "close": 163.65,
            "volume": 134560800,
            "adjusted": 163.65
          }, {
            "date": 15951,
            "open": 165.23,
            "high": 165.58,
            "low": 163.7,
            "close": 164.39,
            "volume": 142322300,
            "adjusted": 164.39
          }, {
            "date": 15952,
            "open": 164.43,
            "high": 166.03,
            "low": 164.13,
            "close": 165.75,
            "volume": 97304000,
            "adjusted": 165.75
          }, {
            "date": 15953,
            "open": 165.85,
            "high": 166.4,
            "low": 165.73,
            "close": 165.96,
            "volume": 62930500,
            "adjusted": 165.96
          }]
        }];
      }
    },

    chart11config: {
      type: Object,
      value: function() {
        return {
          x: function(d) { return d['date'] },
          y: function(d) { return d['close'] },
```

```
          duration: 250,
          margin: {left: 75, bottom: 50}
        }
      }
    },

    chart11JSConfig: {
      type: Function,
      value: function() {
        return function() {
          this._chart.xAxis
            .axisLabel("Dates")
            .tickFormat(function(d) {
                // I didn't feel like changing all the above date values
                // so I hack it to make each value fall on a different date
                return d3.time.format('%x')(new Date(new Date() - (20000 * 86400000)
↪+ (d * 86400000)));
            });
          this._chart.yAxis
            .axisLabel('Stock Price')
            .tickFormat(function(d,i){ return '$' + d3.format(',.1f')(d); });
        }
      }
    }
  }
});
```

The result will look like this:

1.2.10 Demo 7: Using the theme

Make the theme-demo element

In your cell, run:

```
cd src/html/elements
./new-element.js
name of the new element: theme-demo
creating new element <theme-demo>...
removing any .svn folders in  theme-demo
Finished
```

You now have a working *theme-demo* element. We'll edit it soon.

Register the theme-demo element

Edit src/html/elements/elements.html and add the following line

```
<link rel="import" href="theme-demo/theme-demo.html">
```

This will tell AjaXell to load our new element.

Edit the theme-demo element

Edit src/html/elements/theme-demo/theme-demo.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/ts/common-elements/iron-flex-layout-attributes/iron-flex-
↪layout-attributes.html">

<link rel="import" href="/extern/bower_components/paper-button/paper-button.html">
<link rel="import" href="/extern/bower_components/iron-icon/iron-icon.html">
<link rel="import" href="/extern/bower_components/iron-icons/iron-icons.html">
<link rel="import" href="/extern/bower_components/paper-fab/paper-fab.html">

<link rel="import" href="/extern/bower_components/paper-checkbox/paper-checkbox.html">

<link rel="import" href="/extern/bower_components/paper-dropdown-menu/paper-dropdown-
↪menu.html">
<link rel="import" href="/extern/bower_components/paper-material/paper-material.html">
<link rel="import" href="/extern/bower_components/paper-listbox/paper-listbox.html">
<link rel="import" href="/extern/bower_components/paper-menu/paper-menu.html">
<link rel="import" href="/extern/bower_components/paper-item/paper-item.html">
<link rel="import" href="/extern/bower_components/paper-icon-button/paper-icon-button.
↪html">

<link rel="import" href="/extern/bower_components/paper-input/paper-input.html">
<link rel="import" href="/extern/bower_components/paper-input/paper-textarea.html">
<link rel="import" href="/extern/bower_components/paper-radio-button/paper-radio-
↪button.html">
<link rel="import" href="/extern/bower_components/paper-radio-group/paper-radio-group.
↪html">
<link rel="import" href="/extern/bower_components/paper-toggle-button/paper-toggle-
↪button.html">

<link rel="import" href="/extern/bower_components/paper-progress/paper-progress.html">
<link rel="import" href="/extern/bower_components/paper-spinner/paper-spinner.html">

<link rel="import" href="/extern/bower_components/paper-tabs/paper-tabs.html">
<link rel="import" href="/extern/bower_components/paper-tabs/paper-tab.html">

<link rel="import" href="/extern/bower_components/paper-tooltip/paper-tooltip.html">

<link rel="import" href="/extern/bower_components/iron-collapse/iron-collapse.html">
<dom-module id="theme-demo">
  <template>
    <style include="reset-css"></style>
    <style include="iron-flex-layout-attributes"></style>

    <link rel="stylesheet" type="text/css" href="css/theme-demo-min.css?__inline=true
↪">
```

```
40
41      <h1>Buttons</h1>
42      <div horizontal layout>
43        <paper-button>default</paper-button>
44        <paper-button disabled>disabled</paper-button>
45        <paper-button noink>noink</paper-button>
46        <paper-button><iron-icon icon="polymer"></iron-icon> icon</paper-button>
47        <paper-button primary>primary</paper-button>
48        <paper-button secondary>secondary</paper-button>
49        <paper-button info>info</paper-button>
50        <paper-button warning>warning</paper-button>
51        <paper-button error>error</paper-button>
52      </div>
53      <div horizontal layout>
54        <paper-button raised>default</paper-button>
55        <paper-button raised disabled>disabled</paper-button>
56        <paper-button raised noink>noink</paper-button>
57        <paper-button raised><iron-icon icon="polymer"></iron-icon> icon</paper-button>
58        <paper-button raised primary>primary</paper-button>
59        <paper-button raised secondary>secondary</paper-button>
60        <paper-button raised info>info</paper-button>
61        <paper-button raised warning>warning</paper-button>
62        <paper-button raised error>error</paper-button>
63      </div>
64      <div horizontal layout>
65        <paper-button toggles>default</paper-button>
66        <paper-button toggles disabled>disabled</paper-button>
67        <paper-button toggles noink>noink</paper-button>
68        <paper-button toggles><iron-icon icon="polymer"></iron-icon> icon</paper-button>
69        <paper-button toggles primary>primary</paper-button>
70        <paper-button toggles secondary>secondary</paper-button>
71        <paper-button toggles info>info</paper-button>
72        <paper-button toggles warning>warning</paper-button>
73        <paper-button toggles error>error</paper-button>
74      </div>
75      <div horizontal layout>
76        <paper-fab icon="arrow-forward" title="arrow-forward"></paper-fab>
77        <paper-fab disabled icon="create" title="create"></paper-fab>
78        <paper-fab secondary icon="favorite" title="heart"></paper-fab>
79        <paper-fab info mini icon="reply" title="reply"></paper-fab>
80        <paper-fab warning mini icon="reply" title="reply"></paper-fab>
81        <paper-fab error mini icon="reply" title="reply"></paper-fab>
82      </div>
83
84      <h1>Checkboxes</h1>
85      <p>
86        These demonstrate the ability to customize the theme per-panel.
87      </p>
88      <div horizontal layout>
89        <div vertical layout>
90          <paper-checkbox>Default unchecked</paper-checkbox>
91          <paper-checkbox pink>Unchecked with theme override</paper-checkbox>
92          <paper-checkbox checked>Default checked</paper-checkbox>
93          <paper-checkbox pink checked>Checked with theme override</paper-checkbox>
94        </div>
95        <div vertical layout>
96          <paper-checkbox disabled>Default unchecked</paper-checkbox>
97          <paper-checkbox disabled pink>Unchecked with theme override</paper-checkbox>
```

```
        <paper-checkbox disabled checked>Default checked</paper-checkbox>
        <paper-checkbox disabled pink checked>Checked with theme override</paper-
↪checkbox>
      </div>
    </div>

    <h1>Menus</h1>
    <div horizontal layout>
      <paper-dropdown-menu label="Dinosaurs">
        <paper-listbox class="dropdown-content" selected="1">
          <paper-item>allosaurus</paper-item>
          <paper-item>brontosaurus</paper-item>
          <paper-item>carcharodontosaurus</paper-item>
          <paper-item>diplodocus</paper-item>
        </paper-listbox>
      </paper-dropdown-menu>
    </div>
    <div horizontal layout>
      <paper-material elevation="1">
        <paper-listbox>
          <paper-item>Item 1</paper-item>
          <paper-item>Item 2</paper-item>
        </paper-listbox>
      </paper-material>
      <paper-material elevation="1">
        <paper-listbox selected="0">
          <paper-item>Item 1</paper-item>
          <paper-item>Item 2</paper-item>
        </paper-listbox>
      </paper-material>
      <paper-material elevation="1">
        <paper-listbox multi>
          <paper-item>Item 1</paper-item>
          <paper-item>Item 2</paper-item>
        </paper-listbox>
      </paper-material>
    </div>
    <div horizontal layout>
      <paper-material elevation="1">
        <paper-menu selected="0">
          <paper-item>Item 1</paper-item>
          <paper-item>Item 2</paper-item>
        </paper-menu>
      </paper-material>
      <paper-material elevation="1">
        <paper-menu multi>
          <paper-item>Item 1</paper-item>
          <paper-item>Item 2</paper-item>
        </paper-menu>
      </paper-material>
      <paper-menu-button>
        <paper-icon-button icon="menu" class="dropdown-trigger"></paper-icon-button>
        <paper-menu class="dropdown-content">
          <paper-item>Share</paper-item>
          <paper-item>Settings</paper-item>
          <paper-item>Help</paper-item>
        </paper-menu>
      </paper-menu-button>
```

```
155     </div>
156
157     <h1>Input</h1>
158     <div vertical layout>
159       <paper-input label="text input"></paper-input>
160       <paper-textarea label="autoresizing textarea input"></paper-textarea>
161       <paper-input label="disabled input" disabled></paper-input>
162       <paper-input label="input with at most 10 characters" char-counter maxlength="10
    ↪"></paper-input>
163       <paper-input label="this label never floats" no-label-float></paper-input>
164       <paper-input label="this label is always floating" always-float-label␣
    ↪placeholder="placeholder text"></paper-input>
165       <paper-input label="total" type="number">
166         <div prefix>$</div>
167       </paper-input>
168       <paper-slider min="10" max="200" value="110"></paper-slider>
169       <paper-slider min="10" max="200" value="110" pin max-markers="100"></paper-
    ↪slider>
170     </div>
171     <div horizontal layout>
172       <paper-radio-button>Radio</paper-radio-button>
173       <paper-radio-button checked>Radio</paper-radio-button>
174       <paper-radio-button disabled>Disabled</paper-radio-button>
175       <paper-radio-button noink>No ink</paper-radio-button>
176       <paper-radio-button primary>Primary</paper-radio-button>
177       <paper-radio-button secondary>Secondary</paper-radio-button>
178       <paper-radio-button info>Info</paper-radio-button>
179       <paper-radio-button warning>Warning</paper-radio-button>
180       <paper-radio-button error>Error</paper-radio-button>
181     </div>
182     <paper-radio-group selected="small" vertical layout>
183       <paper-radio-button name="small">Small</paper-radio-button>
184       <paper-radio-button name="medium">Medium</paper-radio-button>
185       <paper-radio-button name="large">Large</paper-radio-button>
186     </paper-radio-group>
187     <div vertical layout>
188       <paper-toggle-button>Toggle</paper-toggle-button>
189       <paper-toggle-button checked>Toggle</paper-toggle-button>
190       <paper-toggle-button disabled>Disabled</paper-toggle-button>
191       <paper-toggle-button noink>No ink</paper-toggle-button>
192     </div>
193
194     <h1>Progress</h1>
195     <paper-progress value="10"></paper-progress>
196     <paper-progress value="10" secondary-progress="30"></paper-progress>
197     <paper-spinner active></paper-spinner>
198
199     <h1>Tabs</h1>
200     <paper-tabs selected="0">
201       <paper-tab>TAB 1</paper-tab>
202       <paper-tab>TAB 2</paper-tab>
203       <paper-tab>TAB 3</paper-tab>
204     </paper-tabs>
205     <paper-tabs selected="0" scrollable>
206       <paper-tab>NUMBER ONE ITEM</paper-tab>
207       <paper-tab>ITEM TWO</paper-tab>
208       <paper-tab>THE THIRD ITEM</paper-tab>
209       <paper-tab>THE ITEM FOUR</paper-tab>
```

```
      <paper-tab>FIFTH</paper-tab>
      <paper-tab>THE SIXTH TAB</paper-tab>
      <paper-tab>NUMBER SEVEN</paper-tab>
      <paper-tab>EIGHT</paper-tab>
      <paper-tab>NUMBER NINE</paper-tab>
      <paper-tab>THE TENTH</paper-tab>
      <paper-tab>THE ITEM ELEVEN</paper-tab>
      <paper-tab>TWELFTH ITEM</paper-tab>
    </paper-tabs>
    <paper-tabs selected="0" align-bottom autoselect>
      <paper-tab>ITEM ONE</paper-tab>
      <paper-tab>ITEM TWO</paper-tab>
      <paper-tab>ITEM THREE</paper-tab>
    </paper-tabs>

    <h1>Tooltips</h1>
    <paper-icon-button id="id_1" icon="favorite" alt="heart"></paper-icon-button>
    <paper-tooltip for="id_1" offset="0">&#10084;&#65038;</paper-tooltip>

    <h1>Ripples</h1>
    <paper-material elevation="1" style="width:200px;height:200px;">
      <paper-ripple></paper-ripple>
    </paper-material>
    <paper-material elevation="1" style="width:400px;height:400px;">
      <paper-ripple></paper-ripple>
    </paper-material>


  </template>
  <script src="javascript/theme-demo-min.js?__inline=true"></script>
</dom-module>
```

Now edit src/html/elements/theme-demo/css/theme-demo.scss

```
:host {
  display: block;
  padding: 1em;
}
[horizontal][layout] {
  margin-bottom: 1em;
}

paper-fab {
  margin-right: 1em;
}

paper-radio-button {
  margin-right: 1em;
}

paper-checkbox[pink] {
  --paper-checkbox-unchecked-color: var(--paper-pink-500);
  --paper-checkbox-unchecked-ink-color: var(--paper-pink-500);
  --paper-checkbox-checked-color: var(--paper-pink-500);
  --paper-checkbox-checked-ink-color: var(--paper-pink-500);
  --paper-checkbox-label-color: var(--paper-pink-500);
}

```

```
paper-material {
  margin-right: 1em;
}

paper-progress {
  margin-bottom: 1em;
}

paper-tabs {
  margin-bottom: 1em;
}
```

Now edit src/html/elements/theme-demo/javascript/theme-demo.js

```
Polymer({
    is: 'theme-demo'
});
```

Now execute Grunt to build our new Polymer element.

```
cd src/html
grunt
```

Make the theme-demo panel

Make a new c++ file /src/common/panels/ThemeDemo.cc

```
#include "subsystem/supervisor/panels/ThemeDemo.h"
#include "ajax/PolymerElement.h"

using namespace subsystempanels;
ThemeDemo::ThemeDemo( tsframework::CellAbstractContext* context, log4cplus::Logger&␣
↪logger)
:tsframework::CellPanel(context,logger) {
  logger_ = log4cplus::Logger::getInstance(logger.getName() +".ThemeDemo");
}

void ThemeDemo::layout(cgicc::Cgicc& cgi) {
  remove();
  add(new ajax::PolymerElement("theme-demo"));
}
```

Make the include/subsystem/supervisor/panels/ThemeDemo.h file.

```
#ifndef _subsystem_supervisor_panels_ThemeDemo_h_
#define _subsystem_supervisor_panels_ThemeDemo_h_

#include "ts/framework/CellPanel.h"
#include "log4cplus/logger.h"
#include "cgicc/Cgicc.h"

namespace subsystempanels {
  class ThemeDemo: public tsframework::CellPanel {
    public:
      ThemeDemo(tsframework::CellAbstractContext* context, log4cplus::Logger& logger);
      void layout(cgicc::Cgicc& cgi);
```

```
  };
}
#endif
```

Register the new class in the Makefile.

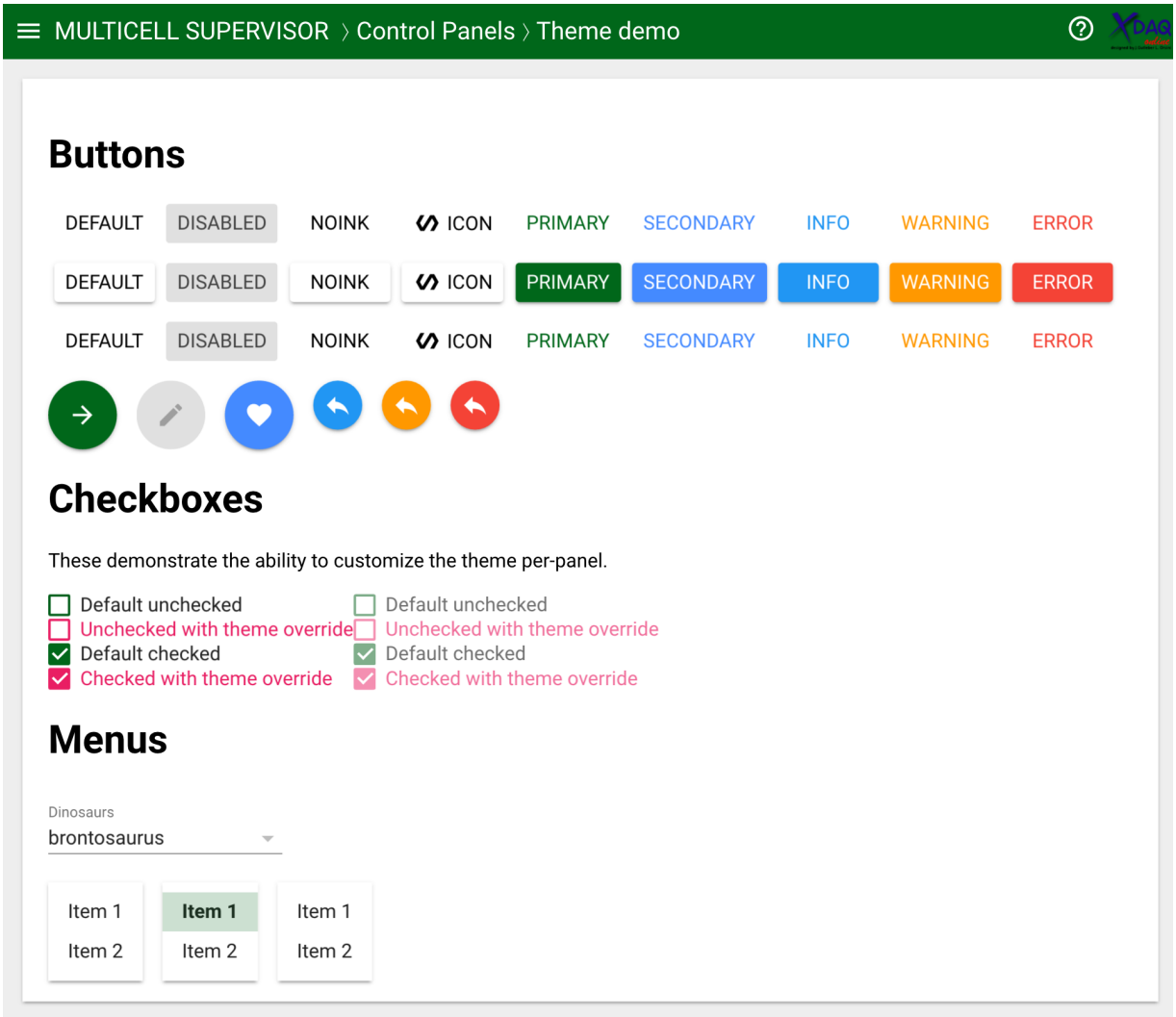
```
Sources=\
    version.cc\
    Cell.cc\
    CellContext.cc\
    Configuration.cc\
    ...
    panels/ThemeDemo.cc \
    ...
```

Now register your new panel in the menu so users can access it.

Edit src/common/Cell.cc

```
#include "subsystem/supervisor/panels/ThemeDemo.h"
...
void subsystemsupervisor::Cell::init() {
  ...
  tsframework::CellPanelFactory* panelF = getContext()->getPanelFactory();
  ...
  panelF->add<subsystempanels::ThemeDemo>("Theme demo");
```

Now you can compile your cell and you should see the ThemeDemo panel in the menu under the 'control-panels' section.



1.2.11 Demo 8: Showing notifications

Make the notifications-demo element

In your cell, run:

```
cd src/html/elements
./new-element.js
name of the new element: notifications-demo
creating new element <notifications-demo>...
removing any .svn folders in  notifications-demo
Finished
```

You now have a working *notifications-demo* element. We'll edit it soon.

Register the notifications-demo element

Edit src/html/elements/elements.html and add the following line

```
<link rel="import" href="notifications-demo/notifications-demo.html">
```

This will tell AjaXell to load our new element.

Edit the notifications-demo element

Edit src/html/elements/notifications-demo/notifications-demo.html

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">
<link rel="import" href="/ts/common-elements/reset-css/reset-css.html">
<link rel="import" href="/ts/common-elements/iron-flex-layout-attributes/iron-flex-
↪layout-attributes.html">
<link rel="import" href="/extern/bower_components/paper-button/paper-button.html">
<link rel="import" href="/ts/ajaxell/html/elements/ag-toaster/throws-toast.html">

<dom-module id="notifications-demo">
  <template>
    <style include="reset-css"></style>
    <style include="iron-flex-layout-attributes"></style>

    <link rel="stylesheet" type="text/css" href="css/notifications-demo-min.css?__
↪inline=true">

    <div horizontal layout>
      <paper-button flex info raised on-click="showInfo">show info</paper-button>
      <paper-button flex info raised on-click="showBigInfo">show info with options</
↪paper-button>
      <paper-button flex info raised on-click="showModalInfo">show modal info</paper-
↪button>
      <paper-button flex info raised on-click="showModalInfox5">show 5 big infos</
↪paper-button>
    </div>

    <div horizontal layout>
      <paper-button flex warning raised on-click="showWarning">show warning</paper-
↪button>
      <paper-button flex warning raised on-click="showBigWarning">show warning with␣
↪options</paper-button>
      <paper-button flex warning raised on-click="showModalWarning">show modal warning
↪</paper-button>
      <paper-button flex warning raised on-click="showModalWarningx5">show 5 big␣
↪warnings</paper-button>
    </div>

    <div horizontal layout>
      <paper-button flex error raised on-click="showError">show error</paper-button>
      <paper-button flex error raised on-click="showBigError">show error with options
↪</paper-button>
      <paper-button flex error raised on-click="showModalError">show modal error</
↪paper-button>
      <paper-button flex error raised on-click="showModalErrorx5">show 5 big errors</
↪paper-button>
    </div>

    <paper-button raised on-click="showLog">show log in console</paper-button>

    <p>
```

```
      Notifications have 3 levels: info, warning, and error.
      If a notification is triggered while another is still visible, different things␣
↪can happen:
    </p>
    <ul>
      <li>
        <p>
          The current notification is a modal window.
        </p>
        <p>
          The new notification will wait for the modal to finish, regardless of the␣
↪level.
        </p>
      </li>
      <li>
        <p>
          The current notification has a lower level than the new notification.
        </p>
        <p>
          The current notification will be discarded immediately and the new␣
↪notification will be shown.
        </p>
      </li>
      <li>
        <p>
          The current notification has an equal of higher level than the new␣
↪notification.
        </p>
        <p>
          The new notification will be queued until the previous one has finished.
          This queue can grow arbitrarily long.
        </p>
      </li>
    </ul>

  </template>
  <script src="javascript/notifications-demo-min.js?__inline=true"></script>
</dom-module>
```

Now edit src/html/elements/notifications-demo/css/notifications-demo.scss

```
:host {
  display: block;
}
[horizontal][layout] {
  margin-bottom: 10px;
}
```

Now edit src/html/elements/notifications-demo/javascript/notifications-demo.js

```
Polymer({
    is: 'notifications-demo',

    behaviors: [throwsToast],

    /**
     * This produces a toast (https://www.google.com/design/spec/components/snackbars-
↪toasts.html)
```

```
     */
    showInfo: function showInfo() {
        this.throwToast({
            'type': 'info',
            'message': 'this is some info at t=' + new Date().getTime(),
            'callback': function callback(response) {
                console.log("callback successfull: ", response);
            }
        });
    },

    /**
     * This produces a toast, but the theme file makes it have an orange background.
     * Also, a warning message has a higher priority than an info message.
     * If an info toast is currently visible and a warning toast is thrown, the info toast will be discarded immediately.
     * if a warning toast is currently visible and an info toast is thrown, the info toast will be displayed after the warning toast has closed.
     * if a warning toast is currently visible and a warning toast is thrown, the warning toast will be displayed after the warning toast has closed.
     */
    showWarning: function showWarning() {
        this.throwToast({
            'type': 'warning',
            'message': 'this is a warning at t=' + new Date().getTime(),
            'callback': function callback(response) {
                console.log("callback successfull: ", response);
            }
        });
    },

    /**
     * This produces a toast, but the theme file makes it have an red background.
     * Also, an error message has a higher priority than an info or warning message.
     */
    showError: function showError() {
        this.throwToast({
            'type': 'error',
            'message': 'this is an error at t=' + new Date().getTime(),
            'callback': function callback(response) {
                console.log("callback successfull: ", response);
            }
        });
    },

    /**
     * This produces a toast. The user can interact with it and click one of two buttons.
     * This is still a toast and timeouts after a few seconds and doesn't force the user
     * to click one of the buttons, the response variable of the callback will be null or
     * the option (type string) the user selected.
     */
    showBigInfo: function showBigInfo() {
        this.throwToast({
            'type': 'info',
            'message': 'this is big info, you can choose something',
```

```
            'options': ['save the world', 'make dinosaurs'],
            'callback': function callback(response) {
                console.log("you have chosen to", response);
            }
        });
    },

    showBigWarning: function showBigWarning() {
        this.throwToast({
            'type': 'warning',
            'message': 'this is a big warning, you will need to choose something',
            'options': ['save the world', 'make dinosaurs'],
            'callback': function callback(response) {
                console.log("you have chosen to", response);
            }
        });
    },

    showBigError: function showBigError() {
        this.throwToast({
            'type': 'error',
            'message': 'this is a big error, you will need to choose something',
            'options': ['save the world', 'make dinosaurs'],
            'callback': function callback(response) {
                console.log("you have chosen to", response);
            }
        });
    },

    /**
     * This produces a modal window. The user must choose one of the provided
     * options and will not be able to do anything else.
     */
    showModalInfo: function showModalInfo() {
        this.throwToast({
            'type': 'info',
            'message': 'this is big info, you will need to choose something',
            'options': ['save the world', 'make dinosaurs'],
            'blocking': true,
            'callback': function callback(response) {
                console.log("you have chosen to", response);
            }
        });
    },

    showModalWarning: function showModalWarning() {
        this.throwToast({
            'type': 'warning',
            'message': 'this is a big warning, you will need to choose something',
            'options': ['save the world', 'make dinosaurs'],
            'blocking': true,
            'callback': function callback(response) {
                console.log("you have chosen to", response);
            }
        });
    },

    showModalError: function showModalError() {
```

```
118         this.throwToast({
119             'type': 'error',
120             'message': 'this is a big error, you will need to choose something',
121             'options': ['save the world', 'make dinosaurs'],
122             'blocking': true,
123             'callback': function callback(response) {
124                 console.log("you have chosen to", response);
125             }
126         });
127     },
128 
129     showModalInfox5: function showModalInfox5() {
130         for (var i = 1; i <= 5; i++) {
131             this.throwToast({
132                 'type': 'info',
133                 'message': 'this is big info ' + i + ', you will need to choose␣
    ↪something',
134                 'options': ['save the world', 'make dinosaurs'],
135                 'blocking': true,
136                 'callback': function callback(response) {
137                     console.log("you have chosen to", response);
138                 }
139             });
140         }
141     },
142 
143     showModalWarningx5: function showModalWarningx5() {
144         for (var i = 1; i <= 5; i++) {
145             this.throwToast({
146                 'type': 'warning',
147                 'message': 'this is big warning ' + i + ', you will need to choose␣
    ↪something',
148                 'options': ['save the world', 'make dinosaurs'],
149                 'blocking': true,
150                 'callback': function callback(response) {
151                     console.log("you have chosen to", response);
152                 }
153             });
154         }
155     },
156 
157     showModalWarningx5: function showModalWarningx5() {
158         for (var i = 1; i <= 5; i++) {
159             this.throwToast({
160                 'type': 'warning',
161                 'message': 'this is big error ' + i + ', you will need to choose␣
    ↪something',
162                 'options': ['save the world', 'make dinosaurs'],
163                 'blocking': true,
164                 'callback': function callback(response) {
165                     console.log("you have chosen to", response);
166                 }
167             });
168         }
169     },
170 
171     /**
172      * AjaXell keeps logs of every thrown toast. This function dumps this log
```

```
     * into the console window of the browser.
     */
    showLog: function showLog() {
        toaster.showLog();
    }
});
```

Now execute Grunt to build our new Polymer element.

```
cd src/html
grunt
```

Make the notifications-demo panel

Make a new c++ file /src/common/panels/NotificationsDemo.cc

```
#include "subsystem/supervisor/panels/NotificationsDemo.h"
#include "ajax/PolymerElement.h"

using namespace subsystempanels;
NotificationsDemo::NotificationsDemo( tsframework::CellAbstractContext* context,␣
↪log4cplus::Logger& logger)
:tsframework::CellPanel(context,logger) {
  logger_ = log4cplus::Logger::getInstance(logger.getName() +".NotificationsDemo");
}

void NotificationsDemo::layout(cgicc::Cgicc& cgi) {
  remove();
  add(new ajax::PolymerElement("notifications-demo"));
}
```

Make the include/subsystem/supervisor/panels/NotificationsDemo.h file.

```
#ifndef _subsystem_supervisor_panels_NotificationsDemo_h_
#define _subsystem_supervisor_panels_NotificationsDemo_h_

#include "ts/framework/CellPanel.h"
#include "log4cplus/logger.h"
#include "cgicc/Cgicc.h"

namespace subsystempanels {
  class NotificationsDemo: public tsframework::CellPanel {
    public:
      NotificationsDemo(tsframework::CellAbstractContext* context, log4cplus::Logger&␣
↪logger);
      void layout(cgicc::Cgicc& cgi);
  };
}
#endif
```

Register the new class in the Makefile.

```
Sources=\
    version.cc\
    Cell.cc\
    CellContext.cc\
```

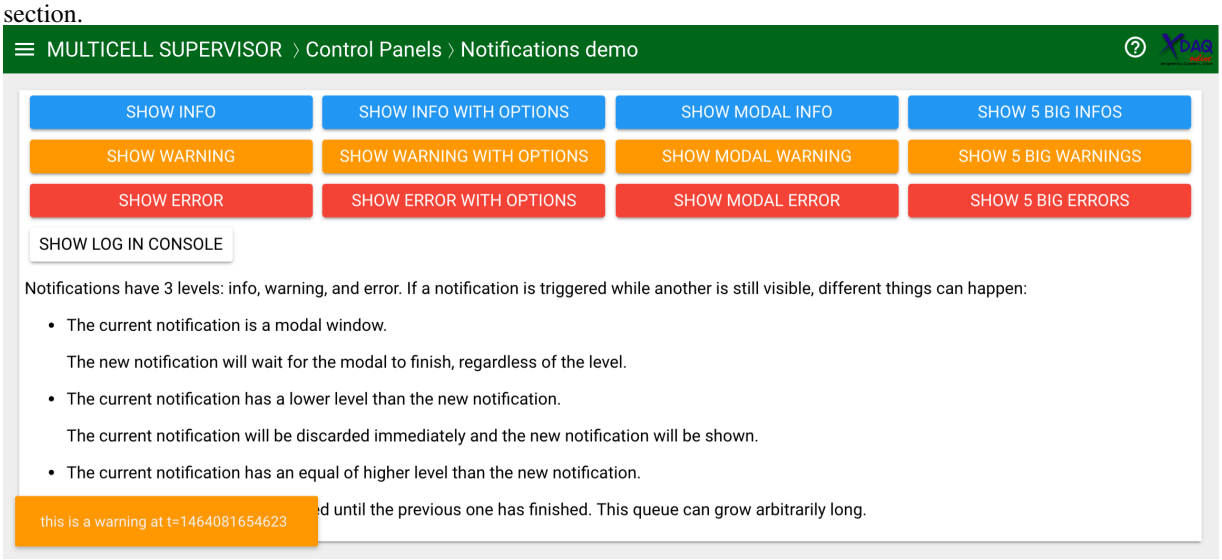
```
    Configuration.cc\
    ...
    panels/NotificationsDemo.cc \
    ...
```

Now register your new panel in the menu so users can access it.

Edit src/common/Cell.cc

```
#include "subsystem/supervisor/panels/NotificationsDemo.h"
...
void subsystemsupervisor::Cell::init() {
  ...
  tsframework::CellPanelFactory* panelF = getContext()->getPanelFactory();
  ...
  panelF->add<subsystempanels::NotificationsDemo>("Notifications demo");
```

Now you can compile your cell and you should see the NotificationsDemo panel in the menu under the 'control-panels' section.



1.3 Advanced section

1.3.1 Writing tests for your elements

Writing tests is important. It will allow you to keep track of your elements as browsers develop. And allows you to detect problems and fix them before they become a big problem.

Some people believe it is a good practice to write your tests before writing actual code. This practice is called Test-Driven Development (TDD). Whether you want to endorse this practice is up to you. But since you made it this far, and are capable of writing complex interfaces, it might be something to consider.

The grunt build tool supports a special command *grunt test*. When executed, it will traverse all your elements and look for a /<element-name>/test/index.html file and run tests defined there.

This chapter will get you started writing tests for this system.

Defining tests in your element

Your elements are defined in /src/html/elements. Every element resides in a subfolder there.

To define tests for your element (in this example called *my-element*), create a file /src/html/elements/my-element/test/index.html with the following content:

/src/html/elements/my-element/test/index.html

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <script src="/bower_components/webcomponentsjs/webcomponents-lite.js"></script>
    <script src="/bower_components/web-component-tester/browser.js"></script>
  </head>
  <body>
    <script>
      WCT.loadSuites([
        'my-element.html',
        'my-element.html?dom=shadow'
      ]);
    </script>
  </body>
</html>
```

This will instruct the testing system to execute tests in *my-element.html* in both shadow DOM mode (=Google Chrome) and shady DOM mode (=Firefox, Safari).

Now for the actual test file:

/src/html/elements/my-element/test/my-element.html

```
<!doctype html>
<html>
  <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, minimum-scale=1.0, initial-
↪scale=1.0, user-scalable=yes">

      <script src="/bower_components/webcomponentsjs/webcomponents-lite.js"></script>
      <script src="/bower_components/web-component-tester/browser.js"></script>

      <link rel="import" href="../my-element.html">
  </head>
  <body>

      <!-- You can use the document as a place to set up your fixtures. -->
      <test-fixture id="my-element-fixture">
          <template>
              <my-element>
                  <h2>my-element</h2>
              </my-element>
          </template>
      </test-fixture>

      <script>
          suite('<my-element>', function() {
              var element;
              setup(function() {
```

```
                element = fixture('my-element-fixture');
            });

            test('defines the "author" property', function() {
                assert.equal(element.someObject.name, 'deinonychus');
            });
            test('distributed children', function() {
                var els = element.getContentChildren();
                assert.equal(els.length, 1, 'one distributed node');
                assert.equal(els[0], element.querySelector('h2'), 'content
↪distributed correctly');
            });
        });
    </script>

  </body>
</html>
```

The first test in this file simply tests if an object called *someObject* defined in your element has a property *name* with value *deinonychus*. It uses the *equal()* function, the one you'll probably use most as well. There are many more functions available to you to write your tests. The library providing these functions is the Chai Assertion Library. Some examples:

```
expect(element).to.contain.all.keys('browserName', 'platform', 'url', 'version');
expect(element).to.not.have.keys('browsers', 'disabled');
expect(element.someObject).to.have.property('name', 'deinonychus');
expect(object1).to.deep.equal(object2);
expect(element.someArray).to.have.length(4);
element.someObject.name.should.be.a('string');
element.someObject.should.have.property('arr').with.length(3);
assert.typeOf(element.someObject.name, 'string');
```

Check http://chaijs.com/ for a full list and documentation.

The second test is more advanced. It tests if your element contains a <content></content> tag. Notice the h2 tag in the template, this code checks if it is actually inserted.

Interacting with your element

You can use the Polymer DOM API to execute JavaScript on elements inside your element (e.g. push buttons):

```
// shorthand function for selection by id
var mybutton = element.$.buttonid;
// or full access to anything with querySelector
// this selects the first paper-button element inside your element
var mybutton = Polymer.dom(element.root).querySelector("paper-button");
mybutton.click();
```

Testing Events

Use addEventListener to respond to events. Do remember to trigger them.

```
test('fires lasers', function(done) {
  element.addEventListener('seed-element-lasers', function(event) {
    assert.equal(event.detail.sound, 'Pew pew!');
```

```
    done();
  });
  element.fireLasers();
});
```

Testing AJAX

Web Component Tester includes Sinon, which enables you to mock XHR requests and create fake servers.

/src/html/elements/my-element/test/my-element.html

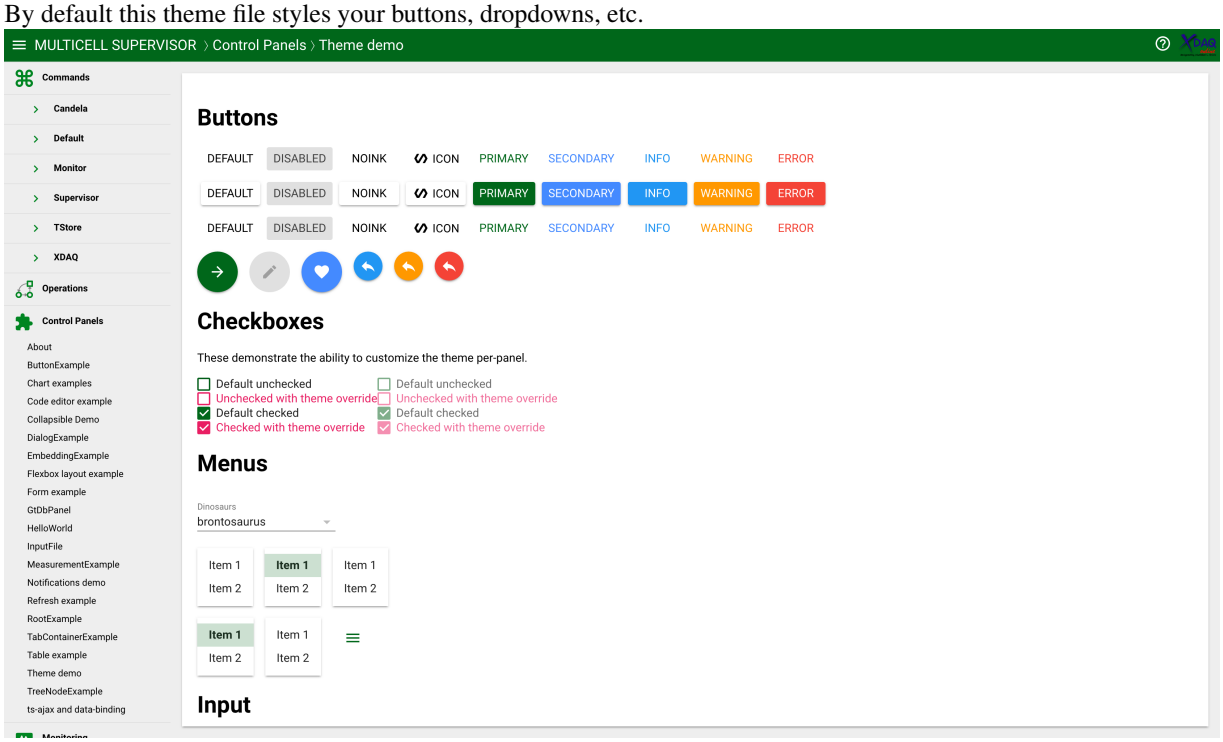
```
<!doctype html>
<html>
  <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, minimum-scale=1.0, initial-
↪scale=1.0, user-scalable=yes">

      <script src="/bower_components/webcomponentsjs/webcomponents-lite.js"></script>
      <script src="/bower_components/web-component-tester/browser.js"></script>

      <link rel="import" href="../my-element.html">
  </head>
  <body>

      <!-- You can use the document as a place to set up your fixtures. -->
      <test-fixture id="my-element-fixture">
          <template>
              <my-element>
                  <h2>my-element</h2>
              </my-element>
          </template>
      </test-fixture>

      <script>
          suite('<my-element>', function() {
              var element;
              var server = sinon.fakeServer.create();
              var responseHeaders = {
                json: { 'Content-Type': 'application/json' }
              };
              setup(function() {
                  element = fixture('my-element-fixture');
                  server.respondWith(
                    'GET',
                    /\/responds_to_get_with_json.*/, [
                      200,
                      responseHeaders.json,
                      '{"success":true}'
                    ]
                  );
              });
              teardown(function() {
                server.restore();
              });

              test('correctly handles the AJAX request', function() {
```

```
                var request = element.functionThatTriggersAnAJAXRequest();
                // catch the response and return fake data
                server.respond();
                expect(request.response).to.be.ok;
                expect(request.response).to.be.an('object');
                expect(request.response.success).to.be.equal(true);
              });
              test('has the correct xhr method', function() {
                var request = ajax.generateRequest();
                expect(request.xhr.method).to.be.equal('GET');
              });
            });
        </script>

    </body>
</html>
```

1.3.2 Theming your cell

The cell follows a theme file. Your elements also follow this by importing the 'reset-css' style element.

By default this theme file styles your buttons, dropdowns, etc.



You will also notice the existence of a *primary* and *secondary* color. By default these colors are green (#00671a) and blue (#448aff) respectively. You can any these things.

Overriding the theme in an element

You can use the CSS styles in your element to override the theme for that particular element.

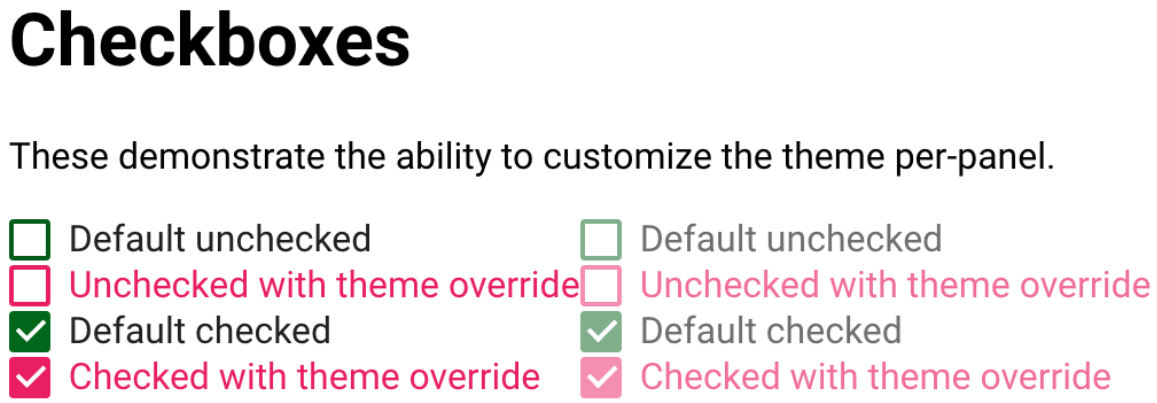
This example will show how you can make pink checkboxes.

If your html code looks like this (notice the *pink* attribute):

```
<paper-checkbox>Default unchecked</paper-checkbox>
<paper-checkbox pink>Unchecked with theme override</paper-checkbox>
```

The theme override can be implemented like this in the CSS:

```
paper-checkbox[pink] {
  --paper-checkbox-unchecked-color: var(--paper-pink-500);
  --paper-checkbox-unchecked-ink-color: var(--paper-pink-500);
  --paper-checkbox-checked-color: var(--paper-pink-500);
  --paper-checkbox-checked-ink-color: var(--paper-pink-500);
  --paper-checkbox-label-color: var(--paper-pink-500);
}
```



Notice that you always have access to the material design (*paper*) colors. They always use the –paper-<colorname>-<intensity> naming convention. A full list can be found at https://www.google.com/design/spec/style/color.html

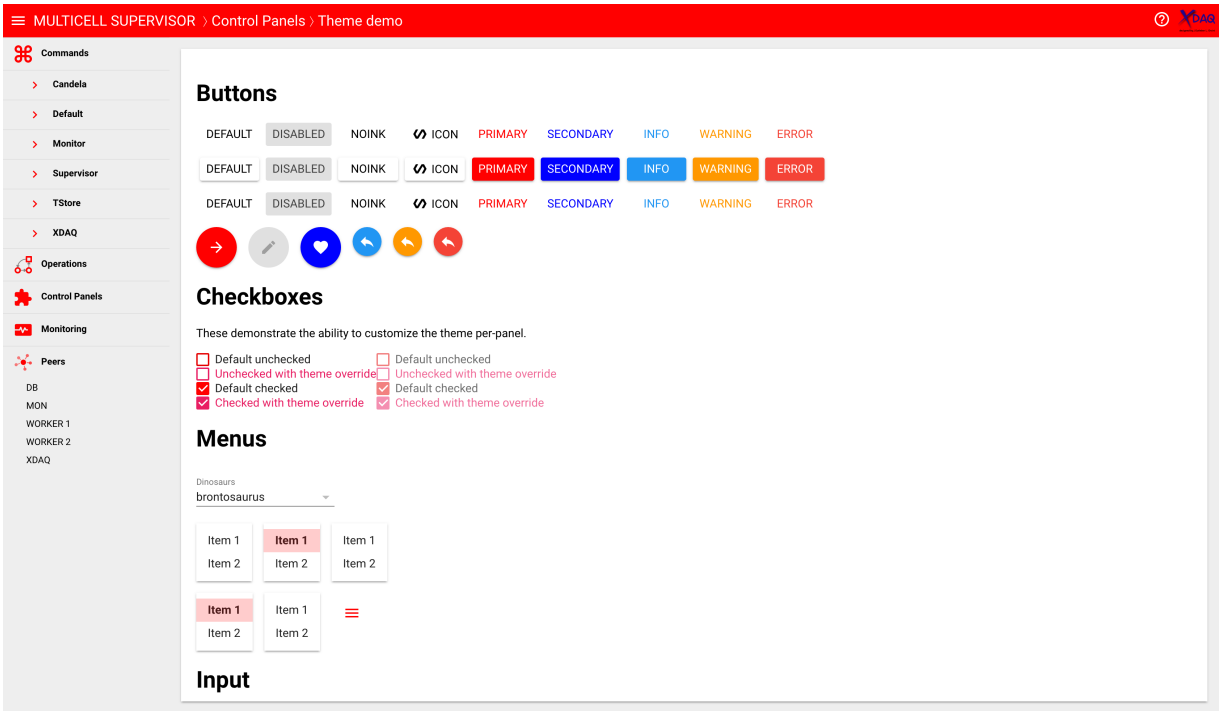
Overriding colors for the entire cell

In the C++ code of your cell you can specify the primary and secondary colors to use in your cell.

In the cell.cc file, add the following lines:

```
// makes primary color red
getContext()->setPrimaryColor(ajax::RGB(255, 0, 0));
// makes secondary color blue
getContext()->setSecondaryColor(ajax::RGB(0, 0, 255));
```

The result will look like this:



Overriding the theme across multiple elements

You can also create your own theme that complements the default theme. A theme file is also a Polymer element, but it lacks JavaScript code.

When creating your own theme file, the HTML should look like this.

```
<link rel="import" href="/extern/bower_components/polymer/polymer.html">

<dom-module id="my-theme">
  <template>
    <link rel="stylesheet" type="text/css" href="css/my-theme-min.css?__inline=true">
  </template>
</dom-module>
```

```
paper-checkbox[pink] {
  --paper-checkbox-unchecked-color: var(--paper-pink-500);
  --paper-checkbox-unchecked-ink-color: var(--paper-pink-500);
  --paper-checkbox-checked-color: var(--paper-pink-500);
  --paper-checkbox-checked-ink-color: var(--paper-pink-500);
  --paper-checkbox-label-color: var(--paper-pink-500);
}
```

The *reset-css* element can be found at https://svnweb.cern.ch/trac/cactus/browser/trunk/cactuscore/ts/common-elements/src/reset-css, the scss file can be consulted to see the full capabilities of theme files.

Available colors

You always have access to the material design (*paper*) colors. They always use the –paper-<colorname>-<intensity> naming convention. A full list can be found at https://www.google.com/design/spec/style/color.html

The framework also defines the following colors:

- –dark-primary-color: #e50000;
- –default-primary-color: #ff0000;
- –primary-color: #ff0000;
- –light-primary-color: #ff0000;
- –secondary-color: #0000ff;
- –primary-color-light-10: #ff0000;
- –primary-color-dark-10: #e50000;
- –secondary-color-light-10: #0000ff;
- –secondary-color-dark-10: #0000e5;
- –primary-color-light-20: #ff0000;
- –primary-color-dark-20: #cc0000;
- –secondary-color-light-20: #0000ff;
- –secondary-color-dark-20: #0000cc;
- –primary-color-light-30: #ff0000;
- –primary-color-dark-30: #b20000;
- –secondary-color-light-30: #0000ff;
- –secondary-color-dark-30: #0000b2;
- –primary-color-light-40: #ff0000;
- –primary-color-dark-40: #990000;
- –secondary-color-light-40: #0000ff;
- –secondary-color-dark-40: #000099;
- –primary-color-light-50: #ff0000;
- –primary-color-dark-50: #7f0000;
- –secondary-color-light-50: #0000ff;
- –secondary-color-dark-50: #00007f;
- –primary-color-light-60: #ff0000;
- –primary-color-dark-60: #660000;
- –secondary-color-light-60: #0000ff;
- –secondary-color-dark-60: #000066;
- –text-primary-color: #ffffff;
- –accent-color: #ff4081;
- –primary-background-color: #ffffff;
- –primary-text-color: #212121;
- –secondary-text-color: #757575;
- –disabled-text-color: #bdbdbd;

- –divider-color: #e0e0e0;
- –error-color: red;

variables and mixins

The colors described earlier are variables. You can use them in your CSS code like this:

```
:host {
  // declare variables
  --my-own-color: green;
  --my-own-padding: 5px;
}
p {
  // use variables
  color: var(--my-own-color);
  padding: var(--my-own-padding);
  // a second argument is a value to use if the variable isn't defined
  background-color: var(--primary-color-light-50, 'blue');
}
```

Mixins are a way to not just store one variable, but an entire block of CSS code.

```
:host {
  // declare mixin
  --my-mixin-name: #{'{
    background-color: green;
    border-radius: 4px;
    border: 1px solid gray;
  }'};
}
p {
  // use mixin
  @apply(--my-mixin-name);
}
```

This can be used for sharing style across elements. You can also put an @apply(); or var(); rule in your CSS without declaring the variable or mixin. This will allow other developers to influence the style of your element in a controlled manner without needing to change the element's source code.

1.4 Available resources

1.4.1 The bower-components package

The bower-components package contains any front-end package that is pulled from the internet (i.e. not made by us).

It currently contains:

- The Polymer library
- iron-elements
- paper-elements
- gold-elements
- neon-elements

- platinum-elements
- juicy-jsoneditor
- juicy-ace-editor
- vaadin-grid
- paper-datatable
- jQuery 2.2.2
- moment.js
- page.js
- cytoscape.js
- file-saver.js
- saveSvgAsPng
- KaTeX

iron-elements

The iron-elements are a set of web components aiming to provide a basic set of tools and enhancements to standard elements, for example to provide them with data-binding capabilities.

These elements do not make assumptions about the used layout or styling, and are expected to maintain a spartan view, if they render a view at all.

Iron-elements aim to extend basic html elements (e.g. <iron-input> to extend <input>), provide façade elements for javascript functionality (e.g. <iron-ajax> to easily make AJAX requests), or provide new functionality that would be considered basic functionality (e.g. <iron-icon> to display an icon).

paper-elements

Paper-elements is a set of elements that focus on bringing Material Designcite{materialdesign} to web components.

Paper-elements aims to extend iron-elements with material design (e.g. <iron-input> becomes <paper-input>), and introduce new elements that are unique to material design (e.g. <paper-toast>)

gold-elements

Gold elements are input elements for specific use cases (e.g. email, phone numbers, credit card numbers, ldots).

They all extend the *paper-input* element and provide specific validation and formatting functionality.

platinum-elements

Platinum-elements are a set of Web Components focused on providing a façade for web-app capabilities like Service Workers, server push, and bluetooth connectivity.

neon-elements

neon-elements are a set of Web Components designed to be façades for the JavaScript animation API to make them available by purely writing HTML.

These elements do not use CSS Transitions, CSS Animations, or SVG, rather they use the new Web Animations API (url{https://www.w3.org/TR/web-animations/}).

These are among the most advanced Web Components in the packages available to panel developers. More info about their usage is provided here: url{https://youtu.be/-tX0e29GQa4}.

juicy-jsoneditor

juicy-json-editor is a web-based tool to view, edit, format, and validate JSON. It has various modes such as a tree editor, a code editor, and a plain text editor.

juicy-ace-editor

juicy-ace-editor is a web component that provides easy access to the Ace library. Ace is an embeddable code editor written in JavaScript. It matches the features and performance of native editors such as Sublime, Vim and TextMate. It can be easily embedded in any web page and JavaScript application. Ace is maintained as the primary editor for Cloud9 IDE and is the successor of the Mozilla Skywriter (Bespin) project.

vaadin-grid

Vaadin Grid is a fully featured datagrid for showing table data. It performs great even with huge data sets, fully supporting paging and lazy loading from any data source like a REST API. Grid allows you sort and filter data and customize how each cell gets rendered.

paper-datatable

A material design implementation of a data table. Currently none of the panels use paper-datatable, and use vaadin-grid instead. This because the development on this element seems dead.

moment.js

Moment.js is a JavaScript library that makes parsing, validating, manipulating, and displaying dates in JavaScript easy.

page.js

Tiny Express-inspired client-side router. It is used by AjaXell to manage the loading of panels.

cytoscape.js

A JavaScript library designed to paint network graphs.

file-saver.js

FileSaver.js implements the HTML5 W3C saveAs() FileSaver interface in browsers that do not natively support it. There is a FileSaver.js demo that demonstrates saving various media types.

FileSaver.js is the solution to saving files on the client-side, and is perfect for webapps that need to generate files, or for saving sensitive information that shouldn't be sent to an external server.

saveSvgAsPng

A JavaScript library that can save an SVG element to a PNG file.

KaTeX

KaTeX is a fast, easy-to-use JavaScript library for TeX math rendering on the web.

1.4.2 The common-elements package

The common-elements package contains web components that are made in-house and are useful across multiple projects (e.g. chart elements).

For more information visit the common-elements page

1.4.3 other packages

Every package can contain its own set of custom-made elements, yours can too.

AjaXell

AjaXell contains a set of elements to make its page-flow work, such as the *page-handler* and the *ts-session* element.

A notable element is *ag-toaster*, which allows anyone to show notifications to the user from their panels. For more information visit the AjaXell page

TS Framework

The TS Framework contains a set of panels that are always included in a cell (i.e. the about panel).

For more information visit the TS Framework page

CHAPTER

TWO

SPHINX SYNTAX EXAMPLES

2.1 This is a Title

2.1.1 subtitle

subsubtitle

and so on

2.1.2 basic markup

italic text.

bold text.

a link <http://xkcd.com/>.

```
*verbatim code*
```

TODO: a reference

2.1.3 lists

- This is a bulleted list.
- It has two items, the second item uses two lines. (note the indentation)

1. This is a numbered list.
2. It has two items too.
    - sub 1
    - sub 2
3. This is another list
4. Two lists separated only by whitespace are concatenated

2.1.4 terms

term (up to a line of text) Definition of the term, which must be indented

and can even consist of multiple paragraphs

next term Description.

2.1.5 line blocks

These lines are
broken exactly like in
the source file.

2.1.6 tables

Header row, column 1 (header rows optional)	Header 2	Header 3	Header 4
body row 1, column 1	column 2	column 3	column 4
body row 2	...	...	

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

2.1.7 links

This is a paragraph that contains a link.

Lorem ipsum [1] dolor sit amet ... [2]

Lorem ipsum *[Ref]* (page **??**) dolor sit amet.

2.1.8 Source code

basic

This is a normal text paragraph. The next paragraph is a code sample:

```
It is not processed in any way, except
that the indentation is removed.

It can span multiple lines.
```

This is a normal text paragraph again.

[1] Text of the first footnote.
[2] Text of the second footnote.

fancy

```
<h1>code block example</h1>
<auto-update data="{{my_variable}}" callback="cpp_callback" handle-as="text"></auto-update>
<span>{{my_variable}}</span>
```

from external file

```
<!doctype html>
<html>
  <head>
    <title>command-input demo</title>

    <meta name="viewport" content="width=device-width, minimum-scale=1.0, initial-scale=1, user-scalable=yes">
    <meta name="mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-capable" content="yes">

    <script>window.Polymer = window.Polymer || {};window.Polymer.dom = 'shadow';</script>
    <script src="/extern/bower_components/webcomponentsjs/webcomponents-lite.js"></script>

    <link rel="import" href="../command-input.html">
    <style>
      command-input {
        margin: 10px;
      }
    </style>
  </head>
  <body unresolved>
    <template is="dom-bind">
      <command-input name="dinosaurs are great" value="true" type="bool"></command-input><br />
      <command-input name="your_favorite_dinosaur" value="deinonychus" type="string"></command-input><br />
      <command-input name="positive number" value="0" type="unsigned int"></command-input>
      <command-input name="a number" value="0" type="int"></command-input>
      <command-input name="fraction" type="float"></command-input>
      <command-input name="fraction" value="0.1e-4" type="double"></command-input><br />
    </template>
  </body>
</html>
```

2.1.9 custom HTML

2.1.10 Fancy Math

Since Pythagoras, we know that $a^2 + b^2 = c^2$.

$$(a+b)^2 = a^2 + 2ab + b^2$$
$$(a-b)^2 = a^2 - 2ab + b^2$$
$$\begin{aligned}(a+b)^2 &= (a+b)(a+b)\\ &= a^2 + 2ab + b^2\end{aligned}$$

$$y = ax^2 + bx + c \tag{2.1}$$
$$f(x) = x^2 + 2xy + y^2 \tag{2.2}$$

Euler's identity, equation (2.3), was elected one of the most beautiful mathematical formulas.

$$e^{i\pi} + 1 = 0 \tag{2.3}$$

Appendix B

Paper

As part of this master thesis it was required to write a paper on the subject. A copy of this paper is enclosed as an appendix in this document.

This page is intentionally left almost blank

Upgrading the Interface and Developer Tools of the Trigger Supervisor Software Framework of the CMS experiment at CERN

Glenn Dirkx, *Student, KU Leuven, glenn.dirkx@student.kuleuven.be*
Dr. Christos Lazaridis, *University of Wisconsin, Christos.Lazaridis@cern.ch*
Dr. Peter Karsmakers, *KU Leuven, peter.karsmakers@kuleuven.be*

Abstract—The Compact Muon Solenoid (CMS) Trigger Supervisor (TS) is a software framework that has been designed to handle the CMS Level-1 trigger setup, configuration and monitoring during data taking as well as all communications with the main run control of CMS.
The interface consists of a web-based GUI rendered by a back-end C++ framework (AjaXell) and a front-end JavaScript framework (Dojo). These provide developers with the tools they need to to write their own custom control panels.
However, currently there is much frustration with this framework given the age of the Dojo library and the various hacks needed to implement modern use cases.
The task at hand is to renew this library and its developer tools, updating it to use the newest standards and technologies, while maintaining full compatibility with legacy code.

This paper describes the requirements, development process, and changes to this framework that were included in the upgrade from v2.x to v3.x.

Index Terms—CERN, CMS, L1 Trigger, C++, Polymer, Web Components.

1 Introduction

The CMS experiment at the European Organization for Nuclear Research (CERN) consists of many components. One of them is the Level-1 (L1) trigger, designed to filter the enormous amount of data generated by the proton-proton collisions at the experiment (currently around 80TB/s[1]). The Trigger Supervisor (TS) is a software project that aims to control the L1 trigger. This includes setup, configuration, and monitoring before and during during data taking. It allows for controlling various aspects of the L1 trigger using panels in a web interface. This interface is custom built for each use case, although some generic panels exist (commands, operations, ...).

The main software library that facilitates this, is AjaXell. It provides the developer tools to make a custom panel. It does, however, have a few problems.

Firstly, the Dojo library that AjaXell uses to render the panels is old. It misses functionality required for modern use cases and it even starts to break down on modern browsers. For example, a modal dialog does not render in Google Chrome 44 but the transparent background that usually forces the user to make a choice in the modal does render. This effectively blocks the user from doing anything and forces the user to reload the page and start over. Today, when a developer wants to provide new functionality, the solution is to just manually write the HTML and JavaScript.

Secondly, the current state of affairs requires developers to write everything in one big C++ file. Panels consist of many languages (C++, HTML, JavaScript, and CSS) and combined with the fact that much functionality is written manually, this results in very messy and unreadable code. Several existing panels are very difficult to modify because of this.

This paper describes the changes that have been made to the TS to solve these problems.

2 Related Work

The full original design of the Trigger Supervisor framework can be consulted in the PHD thesis of Ildefons Magrans de Abril[2]. This thesis describes both the hardware and software design decisions that were made and provide more detail about the TS in general, as this paper merely describes the operator interface redesign.

It's also recommended to read the Phase II upgrade technical proposal of the CMS experiment[3] as it describes the upgrade from Phase I [4] and its new architecture and ideas that the work described here accompanies.

3 Functional Requirements

Ideally the result would be a new framework, much more powerful than its predecessor, that yet manages to achieve 100% compatibility with legacy code.

The main objectives are cleaner code, better maintainability, better documentation, and easier development.

Making the framework easier to develop on, will invite developers to write more advanced code.

4 Upgrade options

Although in the final stage there will be no more legacy code, old code must still remain functional in the new environment to allow for a smooth transition.

This limits the available options at the back-end. Because of this, only extra code to the existing C++ codebase can be added. Changes are not possible.

On the front-end side there are a bit more possibilities. The only important requirement is that whatever the new code looks like, the old Dojo code must be able to run alongside it.

4.0.1 Dojo v1.10

It is not possible to just upgrade to a new version of Dojo. Currently AjaXell uses Dojo 0.4 and starting from Dojo 0.9 there has been a major API change. Two different versions of Dojo cannot run concurrently since they still share a lot of function calls.

Also this approach would not solve any of the currently existing problems. Interfaces would still have messy code and frustrated developers.

4.0.2 Web Components

Web Components[5][6] are additions to the HTML5 standard. They enable a developer to develop custom HTML tags, the idea is to mitigate the 'div soup' problem[7] where the web application's source code increases exponentially in size as the complexity of the app increases.

This standardizes an approach seen in many modern JavaScript frameworks such as AngularJS, Ember.js, Knockout.js, Dojo, and Backbone.js. These all allow a developer to declare specialized 'elements' in order to make developing a smart web application easier. However, by relying on the Web Components standard it can be safely assumed the problem encountered with Dojo 0.9, which introduced breaking API changes, will not occur again. Despite being a new standard, support for all CERN-supported browsers (firefox ESR 24-current) can be achieved using the webcomponents.js polyfill.

4.0.3 Polymer

Polymer is a relatively new library, built directly on the Web Components standards, developed by Google. It represents the way Google thinks Web Components should be used. The reason Polymer is very useful is that it has the potential to allow us to introduce proper Separation of Concerns (SoC) principles (5.1) to the development environment.

5 The new development environment

5.1 Separation of Concerns

SoC is a design primitive, dictating a modular design of software. This has been implemented in three ways.

Firstly, different syntaxes now are housed in their own files. This allows for significantly less messy code and enables us to implement specific optimizations for each language (for example a CSS pre- and post-processor).

Secondly, the developer is not limited to one source file for each syntax. If circumstances would make some code easier to manage if it is housed across multiple files this is now possible. An example of this would be a panel with multiple specialized sections. Separating these sections will make the code easier to read and maintain.

Thirdly, this approach pushes developers to separate data from markup. This is a very good thing as it causes the code to once again be much more readable. By having the C++ code only produce the necessary data and putting all rendering and interaction on the front-end a developer can also safely replace rendering logic or user interaction flow without having to worry about data generation.

5.2 Build Cycle

Instead of loading all the separated files individually at runtime, they will instead be compiled together at compile-time. This will improve loading speeds. The tool used to do this is Grunt (http://gruntjs.com/), a task runner built on nodeJS that is used to compile, minify, lint, unit-test,... front-end code languages. It has very wide community adoption, which results in a very rich set of tools available for use.

5.3 Code optimization

Now that every code language is housed in specialized files some optimizations can be done on them at compile-time. The main objective of these optimizations is to achieve as much browser compatibility as possible.

5.3.1 JavaScript

In order to ensure compatibility with all required browsers all JavaScript code is transpiled by Babel (https://babeljs.io/). This will ensure that newer syntax, like ECMAScript 2016 (ES7), will be transpiled into a more compatible equivalent.

Also the JavaScript code will be transpiled by UglifyJS (https://github.com/mishoo/UglifyJS). This will implement various code optimizations[8] making the code faster.

5.3.2 CSS

Developers are given the possibility to write SASS code, an extension of the CSS syntax, that will be transpiled into CSS on compile-time using libsass (http://sass-lang.com/libsass).

Also Grunt will automatically add vendor-specific prefixes to CSS properties to maintain the required browser compatibility using a tool called autoprefixer (https://github.com/postcss/autoprefixer).

5.4 Code sharing

Code duplication should be minimized as much as possible. Code that is used frequently is therefore moved to a separate code repository available for anyone to use. These include things like chart libraries, layout frameworks, and some in-house components such as an auto-updater.

6 Documentation

Documentation is something commonly taken too lightly. Fortunately there are some tools not only to make good documentation but also to encourage developers later on to write proper documentation.

Most of the documentation is housed along with the source code itself. The goal is to minimize separation of code and documentation as this easily leads to inconsistencies between code and documentation.

6.1 Inline Documentation

Advantages of inline documentation are the reduced chances for outdated documentation and being able to enrich source code with typed annotations [9].

Source code consists of C++, JavaScript, HTML, and CSS code. The inline documentation described here is applicable to the last three.

The syntax used to document JavaScript code is called JSDocs and is currently at version 3[9][10]. It provides us with a rich set of expressions enabling a developer to write documentation comparable to JavaDoc.

In addition there are specific points in the source code where a developer can provide code examples and extra directives to document HTML and CSS code. This is however a non-standard method since there is no standardized way to inline- document any of the other languages.

6.2 Global level

The global level is the only level where documentation is separated from the source code. This houses documentation aimed to teach users and developers the concepts and ways of thinking regarding this codebase. It teaches developers the basics of the structure they will be developing in and the philosophy behind this structure.

This global documentation level is built using Sphinx (http://www.sphinx-doc.org/) and provides a single point of entry for people looking for documentation and will guide readers to the next level of documentation when they are ready for it.

6.3 Package level

The codebase is composed of a number of packages. Each package automatically generates documentation describing its content and capabilities.

	TS 2.1.0	TS 3.4.0	difference	difference (%)
Loading	44.5ms	112.4ms	+67.9ms	+152.58%
Scripting	1227.6ms	1187.6ms	-40ms	-3,26%
Rendering	29.7ms	171.0ms	+141.3ms	+475,76%
Painting	7.5ms	36.1ms	+28.6ms	+381.33%
Other	106.4ms	335.9ms	+229.5ms	+215,69%
Idle	213.6ms	775.1ms	+561.5ms	+262,87%
Total	**1.63s**	**2.62s**	**+990ms**	**+60,73%**

TABLE 1
Page loading times for TS 2.x and 3.x

This documentation is generated in the Grunt build cycle described earlier. It loads and interprets every component of the package and generates a summary page giving a general overview and pointing to several useful resources for each component such as the documentation on the element level, a link to the code repository, and a link to a live demo of the component if available.

The code it uses to render this documentation is housed in the source code of each component. It gets interpreted by Grunt and is then compiled in the package documentation page.

6.4 Element level

The lowest level of documentation is documentation of individual web components. This level is also auto-documented from the component's source code. But unlike the documentation on the package level, where documentation is generated on compile time, the documentation here is rendered on the fly.

This is done by using a specialized web component called 'iron-component-page' (https://elements.polymer-project.org/elements/iron-component-page). It interprets the source code of the component and comments left by the developer and compiles this into a documentation page.

This documentation provides an overview of all the properties and available calls of this component. It can also provide code examples and even live demos.

7 Results

7.1 Loading times

Table 1 shows an overview of the initial full page loading times for the legacy TS (version 2.1.0) and the new TS (version 3.4.0). That is, a page load from a new browser tab with all caches removed.

This test is performed with the timeline panel of Google Chrome 50.0.2661.86 (64-bit).

It is expected that the TS 3.x has higher values for everything in this table, because it loads two front-end libraries (Dojo & Polymer).

Notable is the decrease of scripting time for the TS 3.x relative to the TS 2.x. This is because Dojo is minified and packaged into one JavaScript file in the TS 3.x release, where as in the TS 2.x release it was not. Also, because

this test is performed in Google Chrome, which has native support for Web Components, very little scripting needs to be done. This result will be different in other browsers like Mozilla Firefox, where Web Components support needs to be emulated. Then again, the lazy loading system largely removes this overhead from the initial page loading time, so only minor differences would be expected here.

Rendering time has increased the most going from TS 2.x to TS 3.x. This makes sense as Polymer renders everything on the front-end, whereas Dojo used to render everything server-side. During initial page load this rendering load is primarily caused by the rendering of the left side menu. The increase of painting time follows the same logic as the rendering time.

Also notable is the increase of idle time. This means that the browser needs to wait for a task to finish before it can start another. This is caused because the TS 3.x loads the default panel after the initial page load. Which means the TS makes extra network request, to fetch an interface panel, right after initializing. This is counted with the initial page load. TS 2.x just shows a blank page, it loads no default panel. Because the browser needs to wait for the extra network requests to finish before it can render the default panel, the idle time goes up by a lot.

In total, the initial page loading time increased with about 60%, which is an acceptable increase given the new TS runs 2 libraries concurrently.

7.2 CPU consumption

Both TS releases have negligible CPU usage when doing a fresh page load, and stay at 0% CPU usage when the user is not interacting with the system.

TS 3.x uses hardware acceleration for it's animations since they are all made using CSS transform properties or using Web Animations[11]. The only exception to this is the 'paper-spinner' element. Which displays a loading animation. The TS 2.x release did not have any animations.

7.3 Memory consumption

The Dojo library of TS 2.x contained memory leaks, and could lead to a web browser using an excessive amount of memory when an interface was used for a long duration of time.

Unfortunately, legacy panels in the new TS still suffer from this memory leak. This is because the circular references causing the memory leak reside in the Dojo library itself, and thus would be impractical to address. Therefore, any interface that included auto-refreshes had the highest priority to be converted to a new TS 3.x interface.

Because TS 3.x uses client-side interface rendering rather than server-side as the TS 2.x did, it uses more memory from the browser.

In TS 3.x the memory used by an interface panel will be released after there is a switch to another panel. It is also known that in TS 2.x the memory consumption grows linearly with the amount of panels loaded by the user.

	Google Chrome	Firefox
TS 2.1.0 (Dojo)	20.051MB	7.06MB
TS 3.4.0 (Dojo + Polymer)	24.564MB	10.96MB
difference	+4.513MB	+3.9MB
difference (%)	**+22.51%**	**+55.24%**

TABLE 2
Memory usage for TS 2.x and 3.x in Mozilla Firefox and Google Chrome

To test the difference in memory consumption, both TS versions were opened in a new tab while memory consumption is monitored. No panels are loaded, the interfaces are just left for 120s. The mean memory consumption in those 120s is then taken as the mean memory consumption for that TS release. The results of this test are shown in table 2.

8 Functionality

TS 3.x has functionally more capabilities for the interface than TS 2.x had. More importantly, the TS interface is now no longer bound to one framework. Any Web Component can be used, and extra functionality can be developed in-house. This unlike TS 2.x where developers were functionally bound to the elements the Dojo developers provided.

This makes TS 3.x far more easy to change, and thus more ready for the future.

9 SDK improvements

The fact that multiple programming languages are no longer placed into one file, but distributed across multiple files, makes the developing an interface panel a lot easier.

The Web Components approach to build interfaces gives developers a set of powerful tools that are easy to use and extend.

10 Developed panels

The Control Panels are a set of custom interfaces, developed for an individual cell. The other panels however occur on every cell. And are upgraded as part of the new TS release.

10.1 Commands

The new commands panel use the 'command-input' element for its input. Making it easily extendible to understand more input types (e.g. vectors). Currently it understands number, int, long, unsigned int, unsigned long, short, unsigned short, string, double, and float input.

10.2 Operations

The TS 2.x operations panel had some problems with auto-updating. The state diagram tended to update very late, if it updated at all. Result data and new available commands usually took more than 10 seconds to show up in the interface.

The new operations panel is now far more responsive. The state diagram is available when clicking on an icon, as it was deemed a waste of space to show it by default.

10.3 Flashlists

The flashlist panels now have a user-configurable auto-update function. The flashlist can deploy custom renderers in the table depending on the data type, for example a date will be shown as relative time (e.g. 9 minutes ago), instead of just showing a time stamp. This list of custom renderers can be extended easily.

11 Conclusion

The main objective was to upgrade the TS to be able to provide more advanced interfaces, and to keep compatibility with legacy interfaces.

The new interface engine has achieved 100% backwards compatibility, while providing a completely new way to develop new interfaces.

This new interface engine and can be easily extended and is ready for any future use-cases as it is built to change. The developers are not bound to the functionality of one framework, rather it is build on open standards and thus ensures maximum compatibility with future technologies.

The interface developers now have internal, semi auto-generated, documentation at their disposal and have an active community on the world wide web to fall back to.

Acknowledgments

I would like to thank the following people for their assistance during this project:

Christos Lazaridis for being a great mentor and for not getting mad when I break the nightlies or even SVN itself.

Alessandro Thea for his advice on how to proceed with implementing new functionalities and his supply of motivation and inspiration.

Evangelos Paradas for his guidance trough the architecture of the TS and pointing me to useful resources.

Simone Bologna for his enthusiasm and patience when finding bugs, and his steady supply of ideas.

Furthermore I would like to express my thanks to the entire Online Software team for the freedom and trust I've been given that allowed this project to get as far as it has.

References

[1] T. C. Collaboration, "The cms experiment at the cern lhc," *Journal of Instrumentation*, vol. 3, no. 08, p. S08004, 2008. [Online]. Available: http://stacks.iop.org/1748-0221/3/i=08/a=S08004

[2] I. M. de Abril and C.-E. Wulz, "The CMS Trigger Supervisor: Control and Hardware Monitoring System of the CMS Level-1 Trigger at CERN," Ph.D. dissertation, Barcelona, Autonoma U., 2008. [Online]. Available: http://cds.cern.ch/record/1446282

[3] D. Abbaneo, A. Bal, P. Bloch, O. Buchmueller, F. Cavallari, A. Dabrowski, P. de Barbaro, I. Fisk, M. Girone, F. Hartmann, J. Hauser, C. Jessop, D. Lange, F. Meijers, C. Seez, W. Smith, *The Compact Muon Solenoid Phase II Upgrade Technical Proposal*. European Organization for Nuclear Research (CERN), 2015.

[4] A. Breskin and R. Voss, *The CERN Large Hadron Collider: Accelerator and Experiments*. Geneva: CERN, 2009.

[5] World Wide Web Consortium (W3C), "Custom elements," https://w3c.github.io/webcomponents/spec/custom/, 2015.

[6] Mozilla Developer Network, "Web components," https://developer.mozilla.org/en-US/docs/Web/Web_Components, 2014.

[7] C. House, "Html5 web components: The solution to div soup?" https://www.pluralsight.com/blog/software-development/html5-web-components-overview, 2015.

[8] M. Bazon, "Uglifyjs the compressor," http://lisperator.net/uglifyjs/compress, 2012.

[9] Google Developers, "Annotating javascript for the closure compiler," https://developers.google.com/closure/compiler/docs/js-for-compiler, 2016.

[10] "@use jsdoc," http://usejsdoc.org/, 2011.

[11] A. D. T. A. Brian Birtles, Shane Stephens, "Web animations," https://w3c.github.io/web-animations/, 2016.

FACULTY OF ENGINEERING TECHNOLOGY
TECHNOLOGY CAMPUS GEEL
Kleinhoefstraat 4
2440 GEEL, Belgium
tel. + 32 14 56 23 10
fax + 32 14 58 48 59
fet.geel@kuleuven.be
www.fet.kuleuven.be